Fear and loathing in Scala and Kotlin interop

by Marharyta Nedzelska, Wix.com

Who am I?

Marharyta Nedzelska

Software Engineer @ Wix

KKUG & KotLand Kyiv

Speaker

https://medium.com/@margoqueen95

@jMargaritaN twitter

KKUG

facebook.com/groups/KyivKUG/

@KyivKUG

https://www.meetup.com/KyivKUG/

Who am I?

Marharyta Nedzelska

Software Engineer @ Wix

KKUG & KotLand Kyiv

Speaker

BOXER!!!

DISCLAIMER

Scala VS Kotlin

DISCLAIMER

AGENDA

01 WHY???

02 WHAT IS INTEROP

03 PROBLEMS

04 TIPS & TRICKS

05 SUMMARY

01

WHY???

WHY?

CAUSE IT'S LEGAL*

*not everywhere

WHAT IS SCALA?

Scala
OOP
FP
What the hell is this?

WHAT IS SCALA?

functional
academic
JVM language

WHAT IS SCALA?

functional
academic
JVM language

WHAT IS SCALA?

**functional
academic
JVM language**

Better JAVA

WHAT IS SCALA?

functional
academic
JVM language

Better JAVA

WHAT IS KOTLIN?

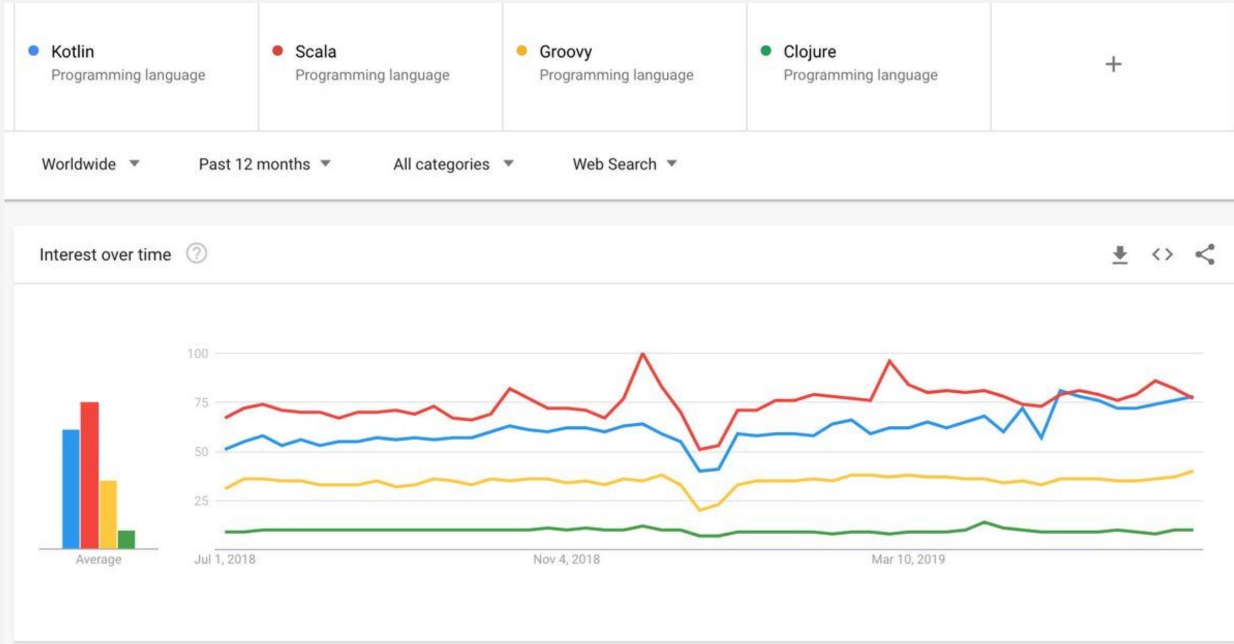
WHAT IS KOTLIN?

pragmatic
language

WHAT IS KOTLIN?

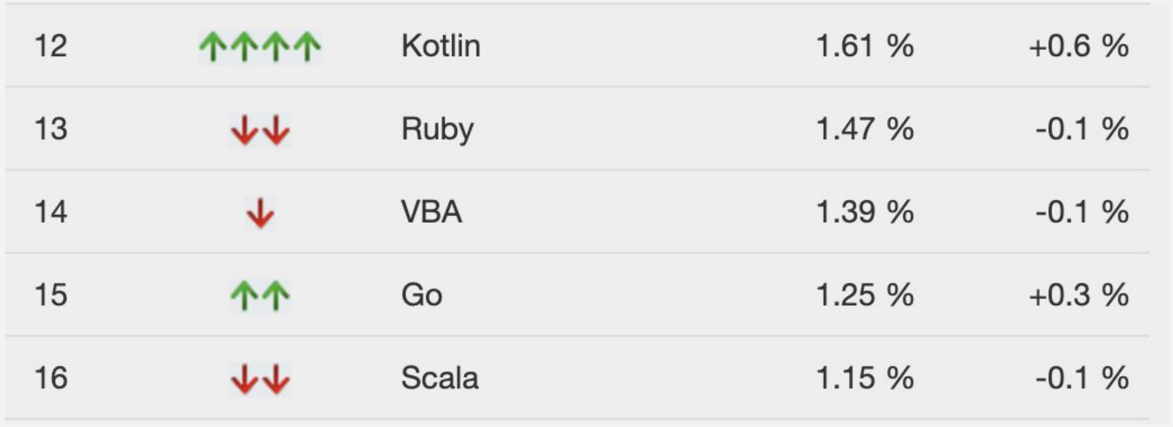
pragmatic
language

WHY?



Kotlin
Programming language
Scala
Programming language
Groovy
Programming language
Clojure
Programming language
+
Worldwide
Past 12 months
All categories
Web Search
Interest over time
100
75
50
25
Average
Jul 1, 2018
Nov 4, 2018
Mar 10, 2019

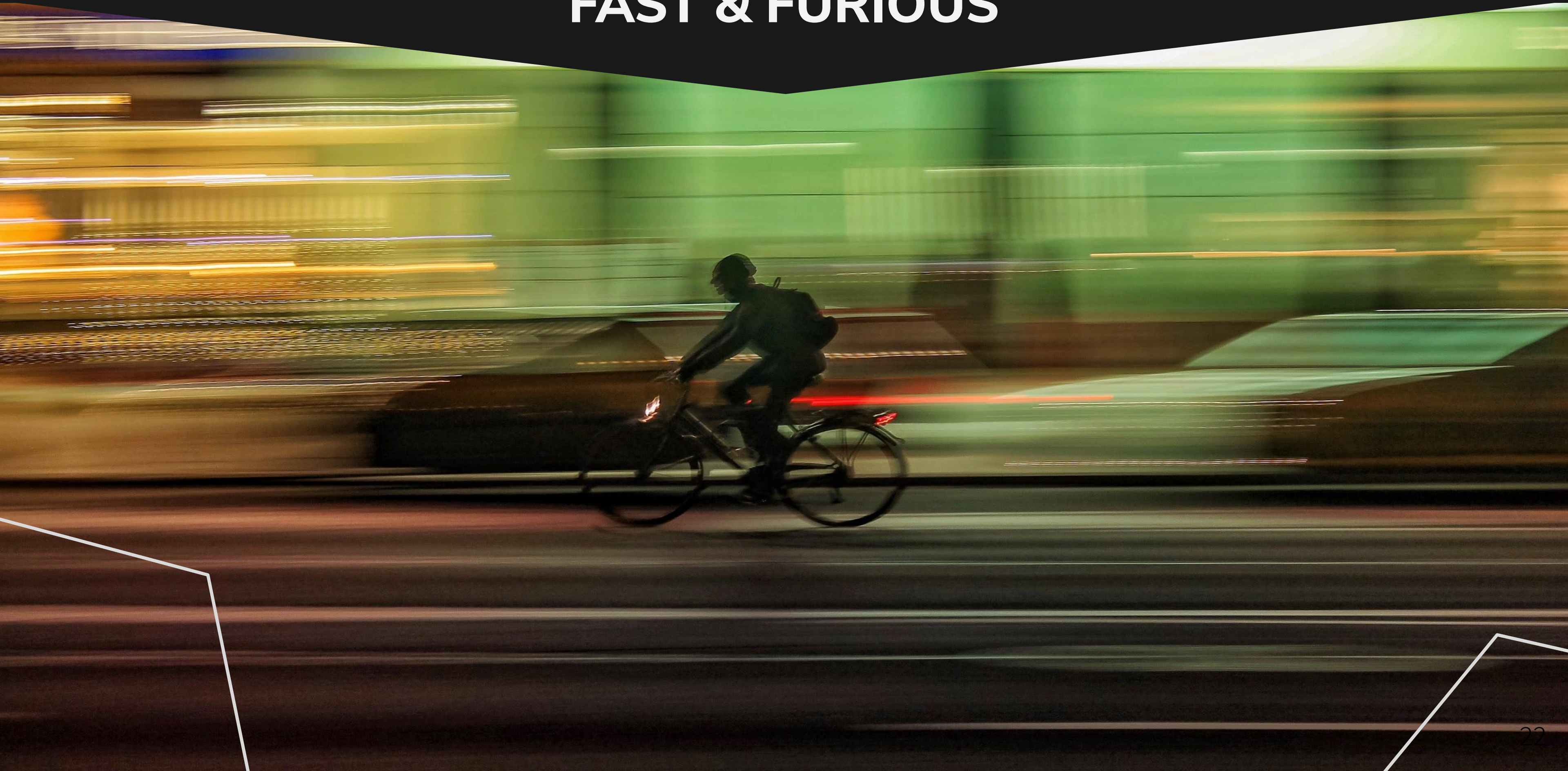
WHY?



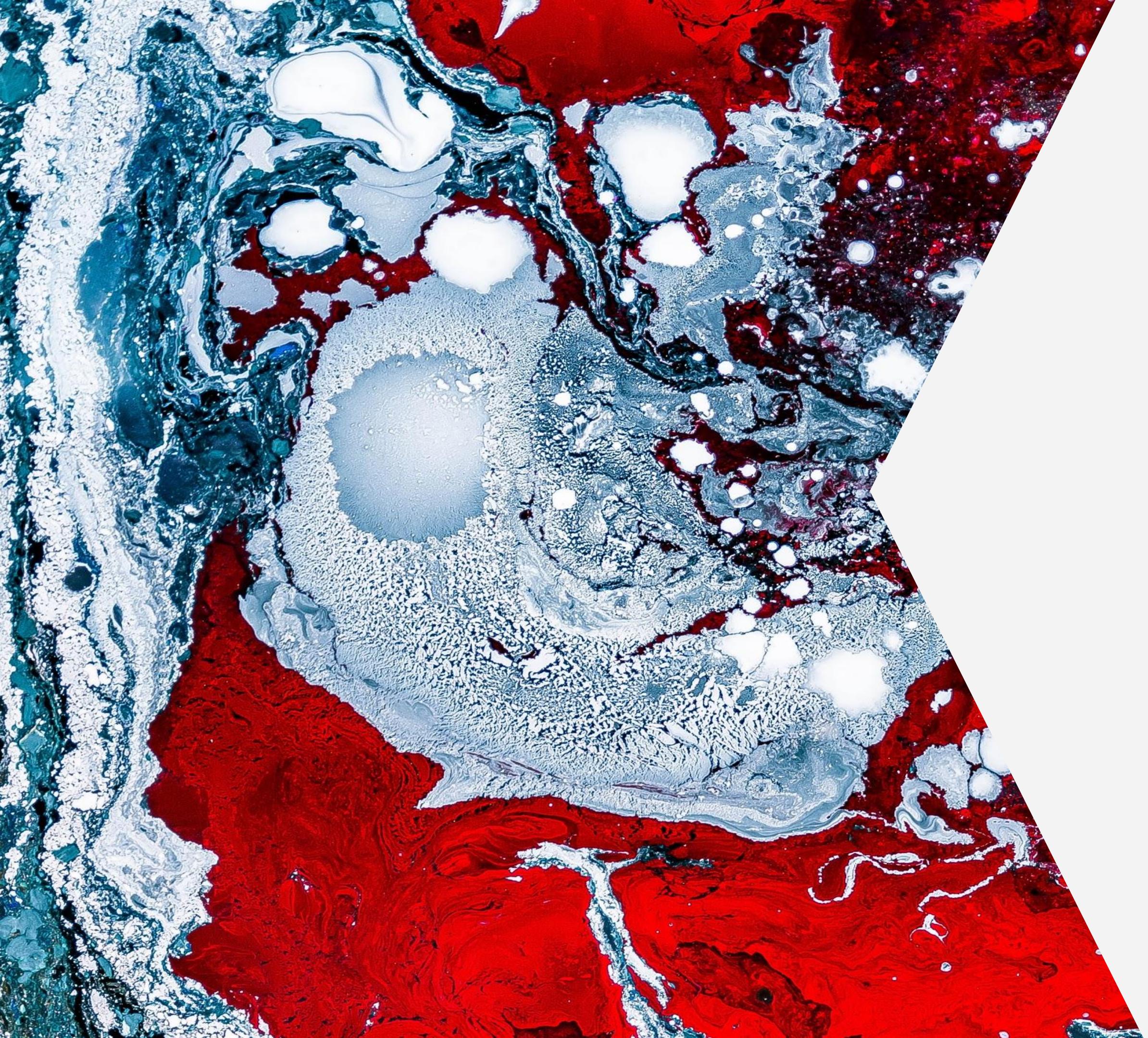
12	↑↑↑↑	Kotlin	1.61 %	+0.6 %
13	↓↓	Ruby	1.47 %	-0.1 %
14	↓	VBA	1.39 %	-0.1 %
15	↑↑	Go	1.25 %	+0.3 %
16	↓↓	Scala	1.15 %	-0.1 %

WHY KOTLIN?

- **Popular**
- **Simple**
- **Attractive for employees**
- **Multiplatform**
- **IDE friendly**
- **...**



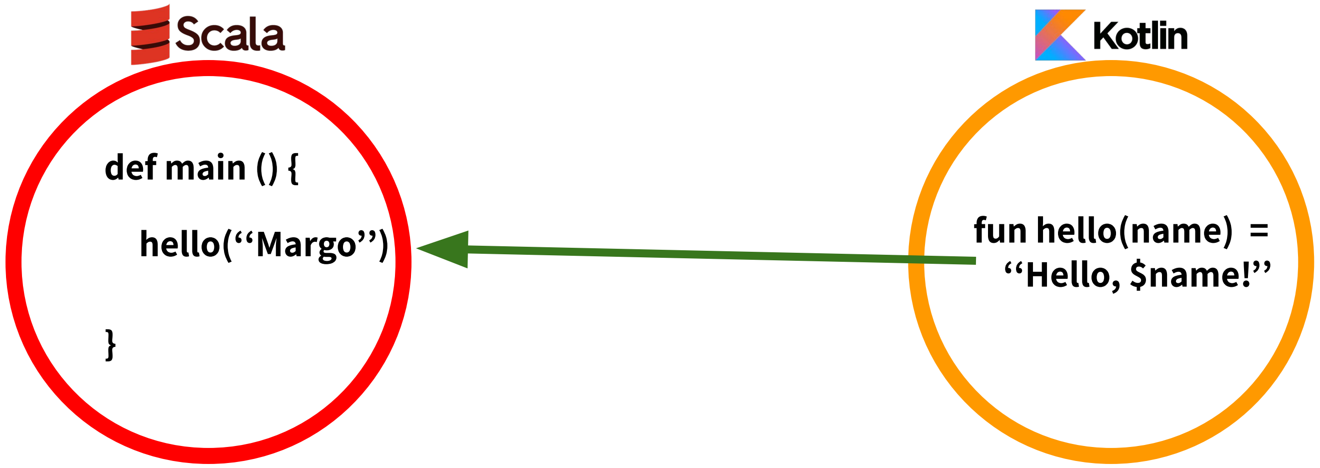
FAST & FURIOUS



02

INTEROP

INTEROPERABILITY(INTEROP)



```
def main () {
    hello(“Margo”)
}
```

```
fun hello(name) =
  “Hello, $name!”
```

HOW CAN WE USE IT?

HOW CAN WE USE IT?

NEWS PAPER
B. Everett Blaylock
Das ander Cevient der Schweiz 220, Jahngang
Zun Eldgen Dank Under Franken Wienee
Das ander Cevient der Schweiz
Das ander Cevient der Schweiz

HOW CAN WE USE IT?

NEWS PAPER
B. Everett Blaylock
Das ander Cevient der Schweiz 220, Jahngang
Zun Eldgen Dank Under Franken Wienee
Das ander Cevient der Schweiz
Das ander Cevient der Schweiz
Scala

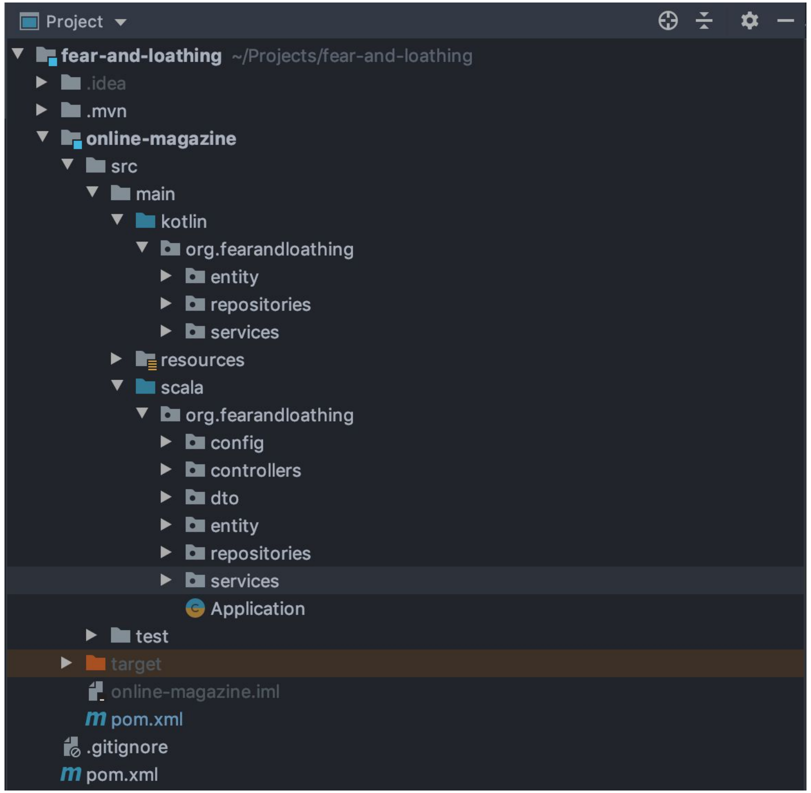
HOW CAN WE USE IT?

NEWS PAPER
B. Everett Blaylock
Das ander Cevient der Schweiz 220, Jahngang
Zun Eldgen Dank Under Franken Wienee
Das ander Cevient der Schweiz
Scala

+ comments

HOW CAN WE USE IT?

LET'S GO



```
Project
▼ fear-and-loathing ~/Projects/fear-and-loathing
  ▶ .idea
  ▶ .mvn
  ▼ online-magazine
    ▼ src
      ▼ main
        ▼ kotlin
          ▼ org.fearandloathing
            ▶ entity
            ▶ repositories
            ▶ services
        ▶ resources
        ▼ scala
          ▼ org.fearandloathing
            ▶ config
            ▶ controllers
            ▶ dto
            ▶ entity
            ▶ repositories
            ▶ services
              Application
      ▶ test
    ▶ target
      online-magazine.iml
      pom.xml
    .gitignore
    pom.xml
```

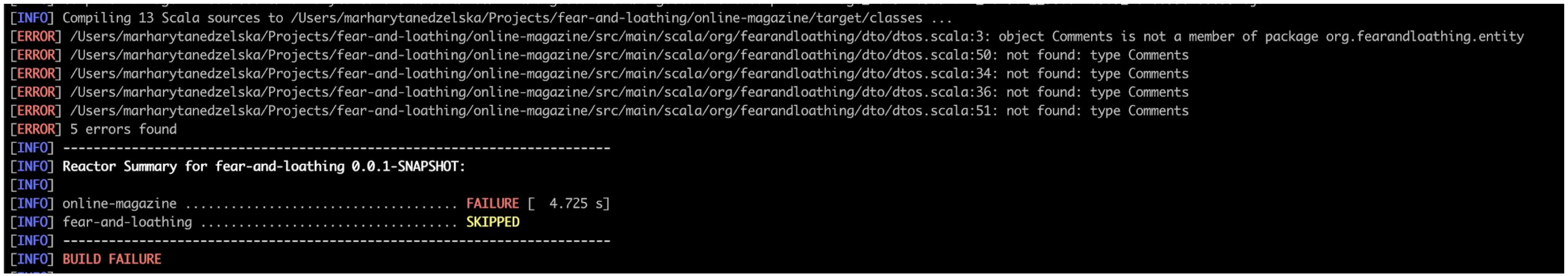
LET'S GO

Scala	2.13.2
Kotlin	1.3.70
Java™	13

LET'S GO

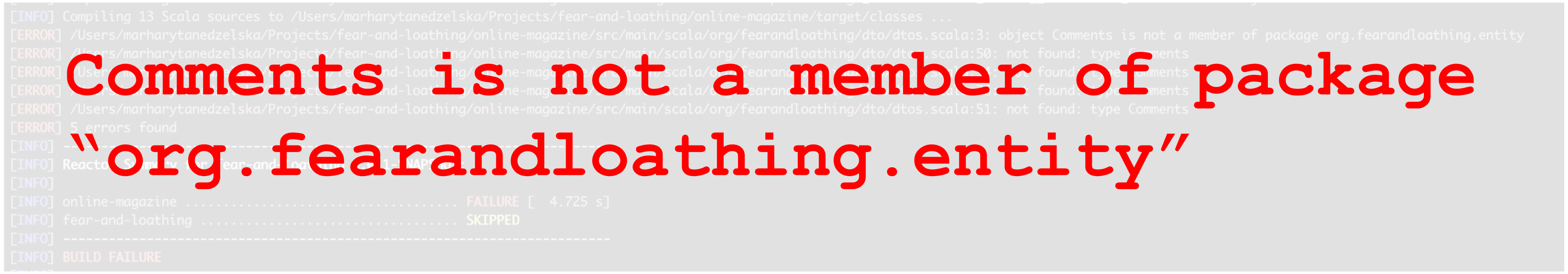
Scala	$2.\boxed{13.}2$
Kotlin	$\boxed{1.3.}70$
Java™	$\boxed{13}$

OOOOOPS!



```
[INFO] Compiling 13 Scala sources to /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/target/classes ...
[ERROR] /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/src/main/scala/org/fearandloathing/dto/dtos.scala:3: object Comments is not a member of package org.fearandloathing.entity
[ERROR] /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/src/main/scala/org/fearandloathing/dto/dtos.scala:50: not found: type Comments
[ERROR] /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/src/main/scala/org/fearandloathing/dto/dtos.scala:34: not found: type Comments
[ERROR] /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/src/main/scala/org/fearandloathing/dto/dtos.scala:36: not found: type Comments
[ERROR] /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/src/main/scala/org/fearandloathing/dto/dtos.scala:51: not found: type Comments
[ERROR] 5 errors found
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for fear-and-loathing 0.0.1-SNAPSHOT:
[INFO]
[INFO] online-magazine .................................... FAILURE [  4.725 s]
[INFO] fear-and-loathing .................................. SKIPPED
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
```

OOOOOPS!



```
[INFO] Compiling 13 Scala sources to /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/target/classes ...
[ERROR] /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/src/main/scala/org/fearandloathing/dto/dtos.scala:3: object Comments is not a member of package org.fearandloathing.entity
[ERROR] /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/src/main/scala/org/fearandloathing/dto/dtos.scala:50: not found: type Comments
[ERROR] [illegible]
[ERROR] [illegible]
[ERROR] /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/src/main/scala/org/fearandloathing/dto/dtos.scala:51: not found: type Comments
[ERROR] 5 errors found
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for fear-and-loathing [illegible]
[INFO]
[INFO] online-magazine ................................... FAILURE [  4.725 s]
[INFO] fear-and-loathing ................................. SKIPPED
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
```

Comments is not a member of package “org.fearandloathing.entity”

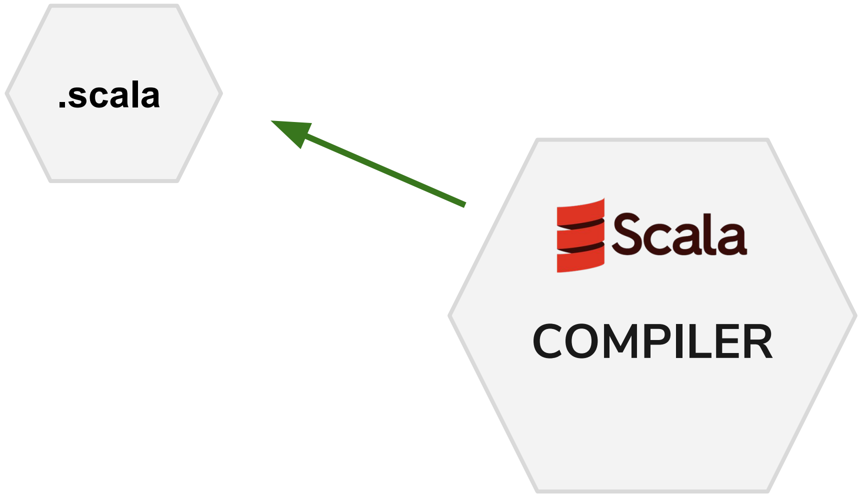
03

PROBLEMS

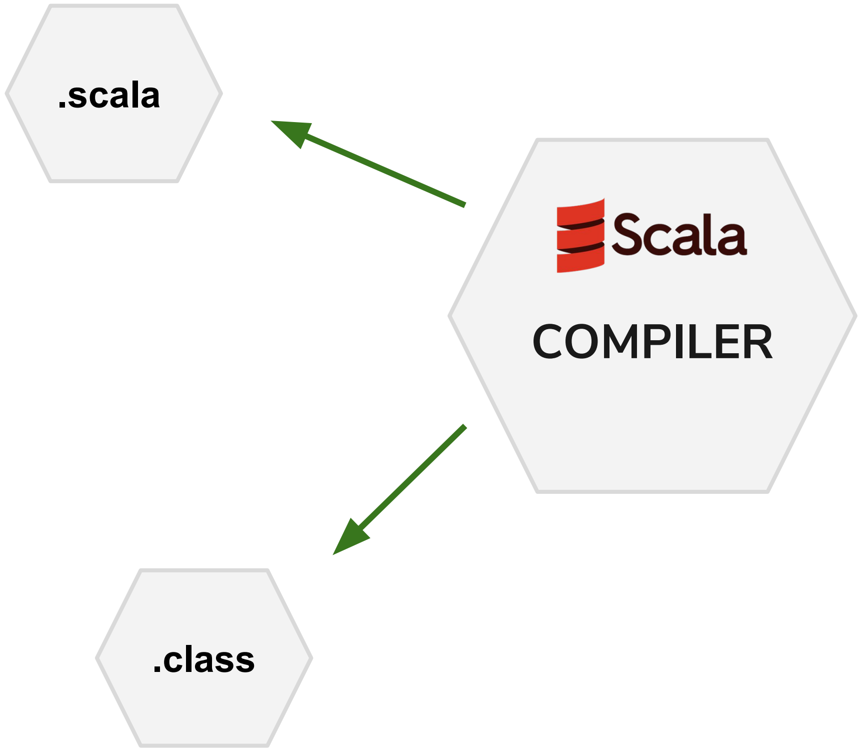
PROBLEM 1

Las Vegas

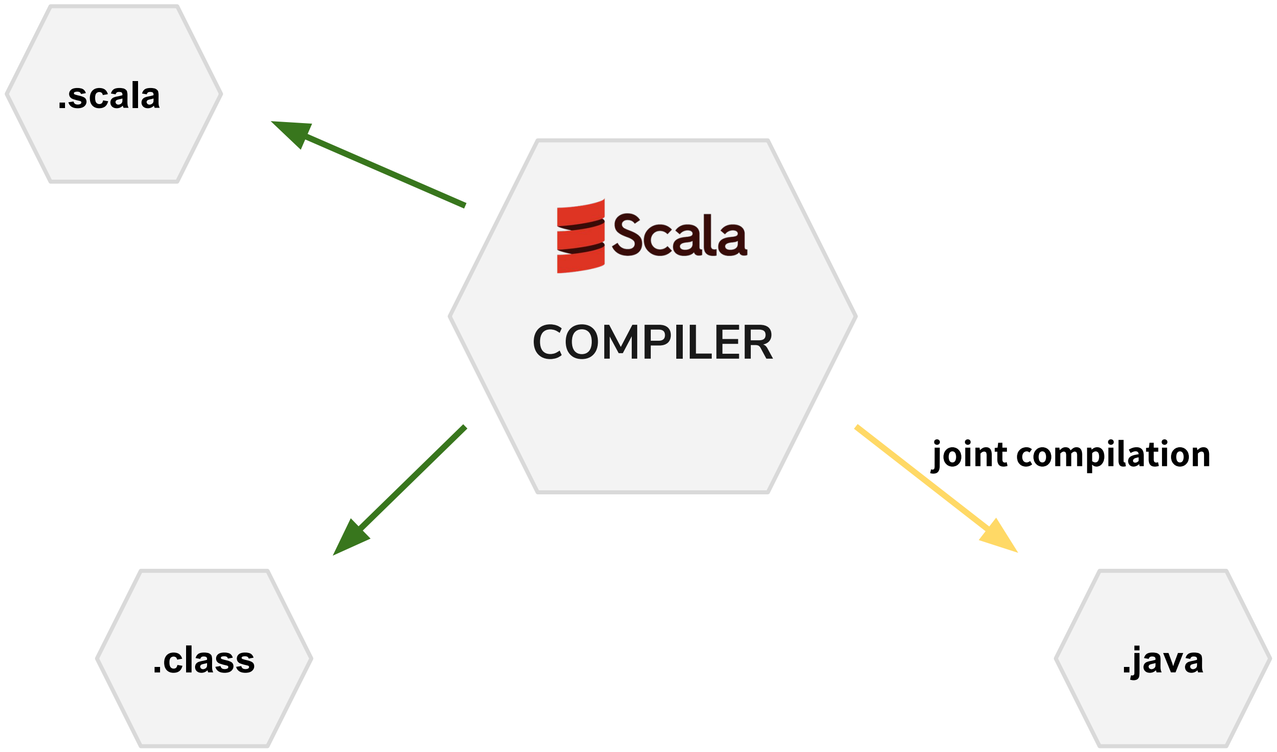
THE REVENGE OF A COMPILER



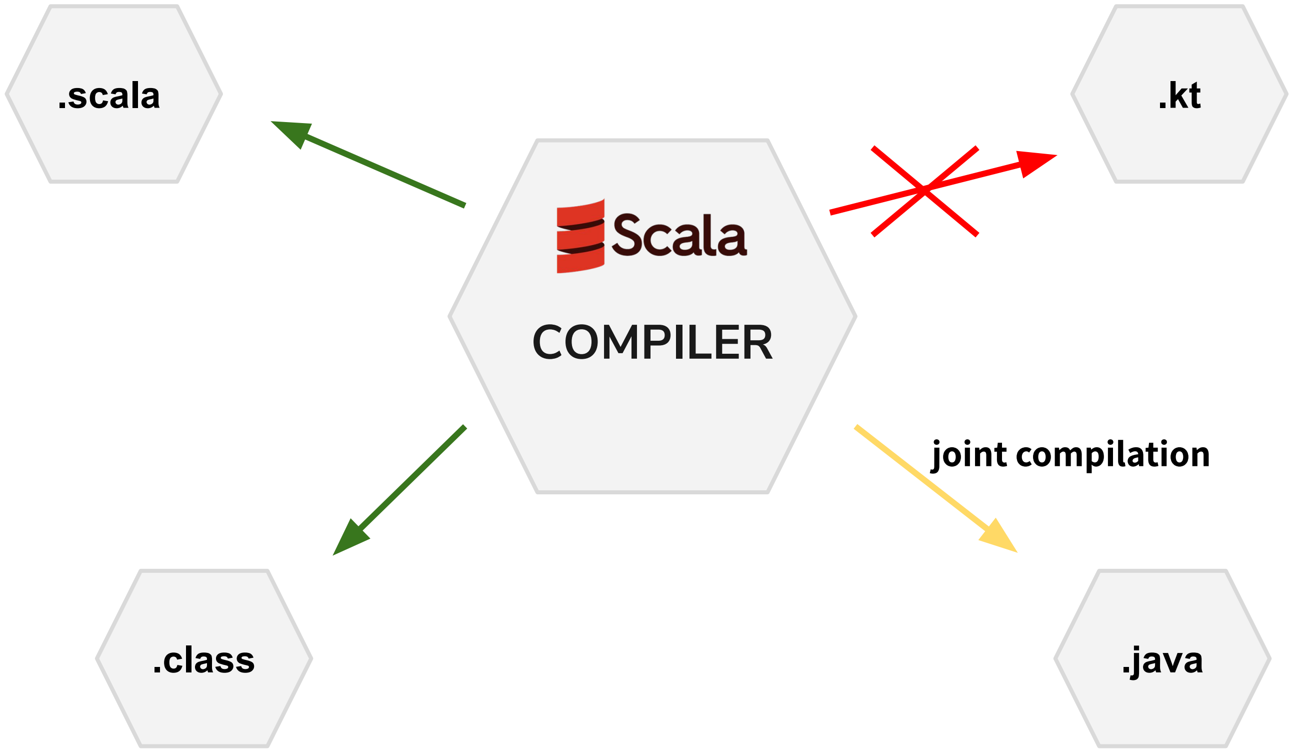
THE REVENGE OF A COMPILER

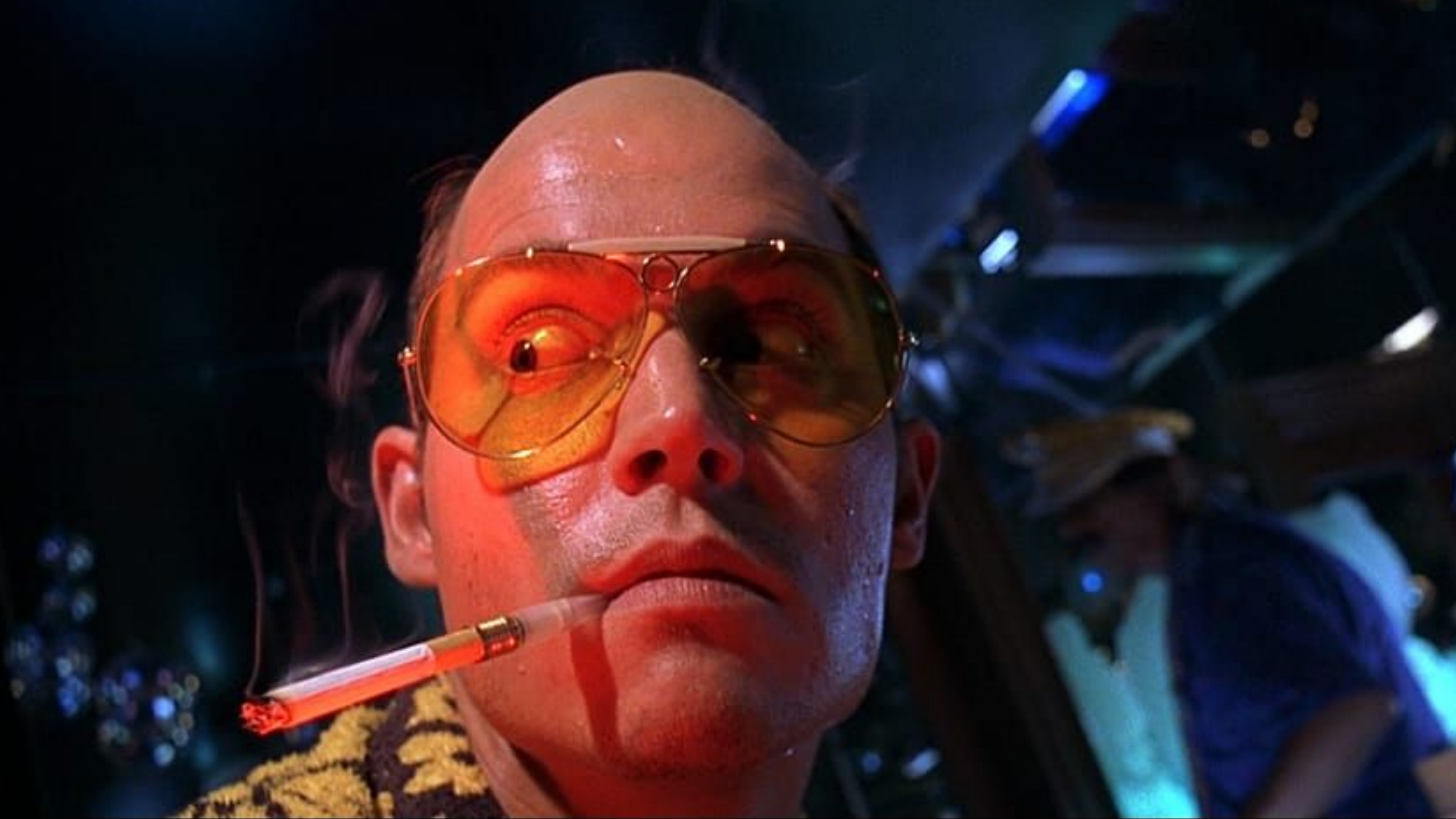


THE REVENGE OF A COMPILER



THE REVENGE OF A COMPILER





THE REVENGE OF A COMPILER

COMPILE KOTLIN FIRST

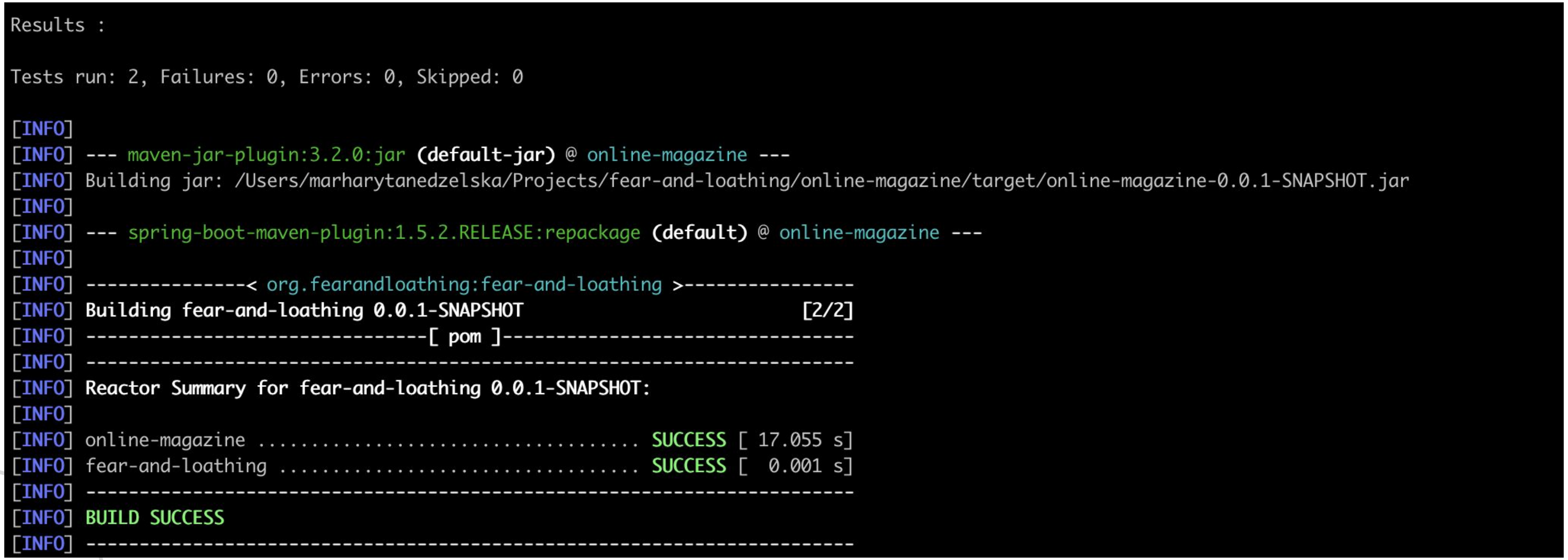
POM.XML

```
<build>
  <plugins>
    <plugin>Scala plugin</plugin>
    <plugin>Kotlin plugin</plugin>
  </plugins>
</build>
```

POM.XML

```
<build>
  <plugins>
    <plugin>Kotlin plugin</plugin>
    <plugin>Scala plugin</plugin>
  </plugins>
</build>
```

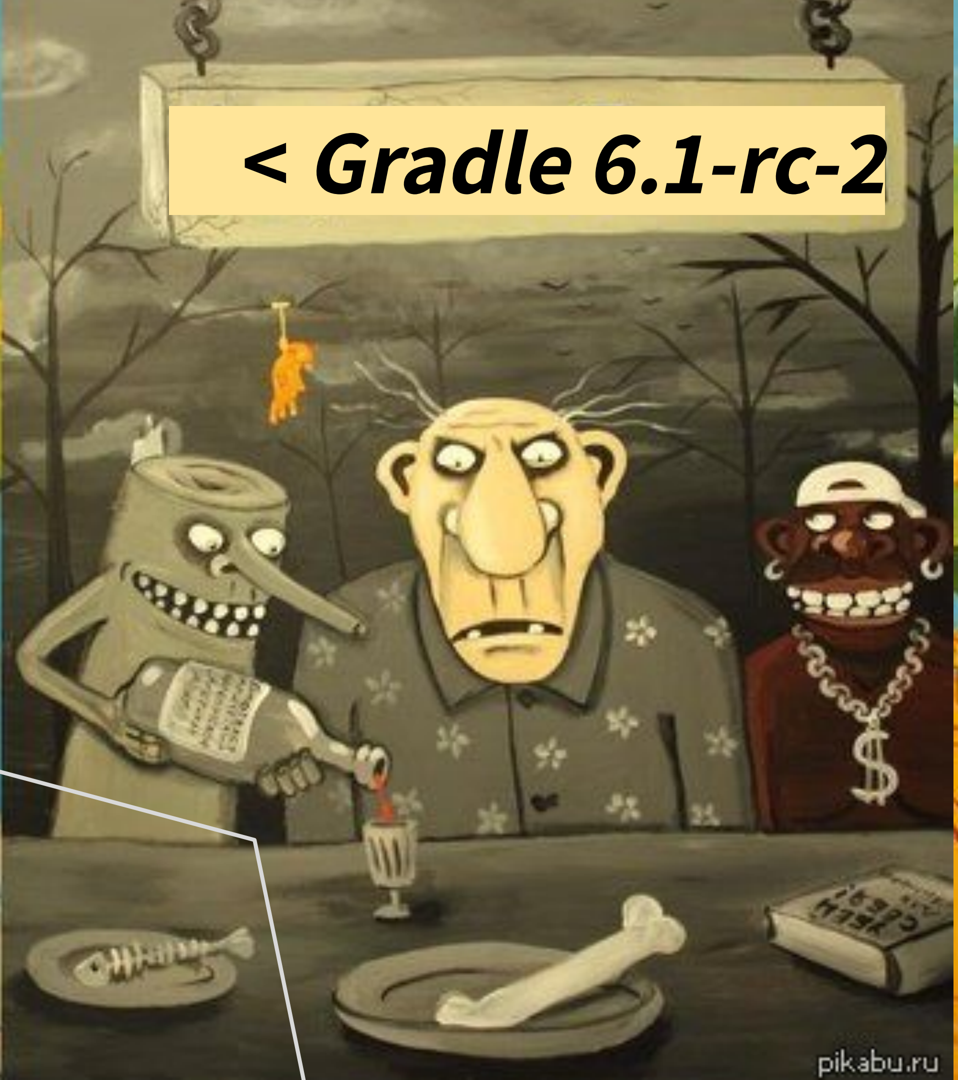
LET'S BUILD

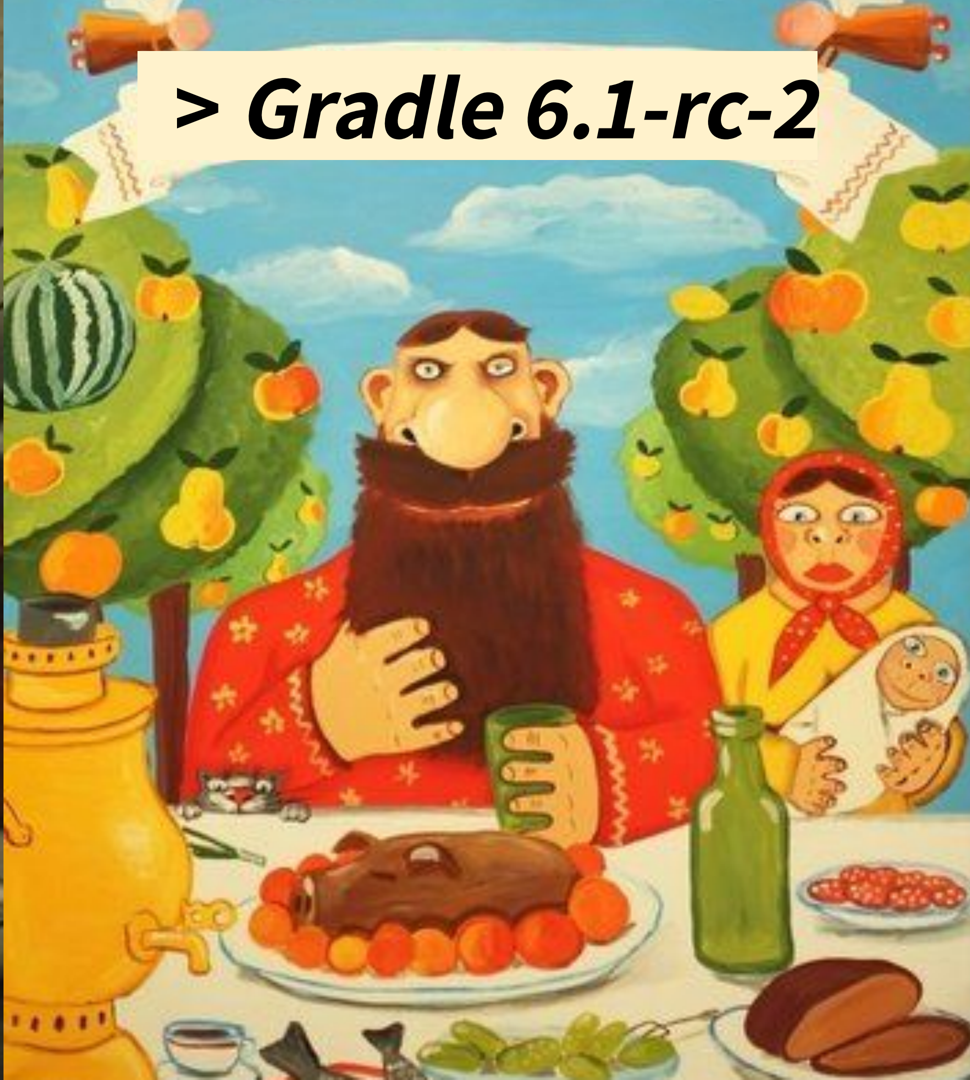


```
Results :

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0

[INFO]
[INFO] --- maven-jar-plugin:3.2.0:jar (default-jar) @ online-magazine ---
[INFO] Building jar: /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/target/online-magazine-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot-maven-plugin:1.5.2.RELEASE:repackage (default) @ online-magazine ---
[INFO]
[INFO] ---------------< org.fearandloathing:fear-and-loathing >----------------
[INFO] Building fear-and-loathing 0.0.1-SNAPSHOT                          [2/2]
[INFO] --------------------------------[ pom ]---------------------------------
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for fear-and-loathing 0.0.1-SNAPSHOT:
[INFO]
[INFO] online-magazine .................................... SUCCESS [ 17.055 s]
[INFO] fear-and-loathing .................................. SUCCESS [  0.001 s]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

THE REVENGE OF A COMPILER

That’s *Maven*!
What about *Gradle*?


< Gradle 6.1-rc-2
pikabu.ru


> Gradle 6.1-rc-2

Gradle 6.1-rc-2

```
tasks.named('compileKotlin') {
    classpath = sourceSets.main.compileClasspath
}
tasks.named('compileScala') {
    classpath += files(sourceSets.main.kotlin.classesDirectory)
}
```

Gradle 6.1-rc-2

```
tasks.named('compileKotlin') {
    classpath = sourceSets.main.compileClasspath
}
tasks.named('compileScala') {
    classpath += files(sourceSets.main.kotlin.classesDirectory)
}
```

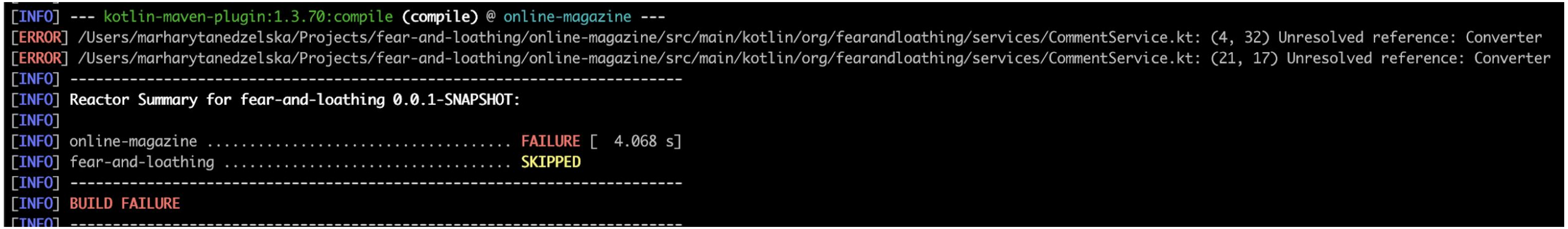
THE REVENGE OF A COMPILER

WHAT IF KOTLIN CODE DEPENDS ON SCALA CODE?

THE REVENGE OF A COMPILER

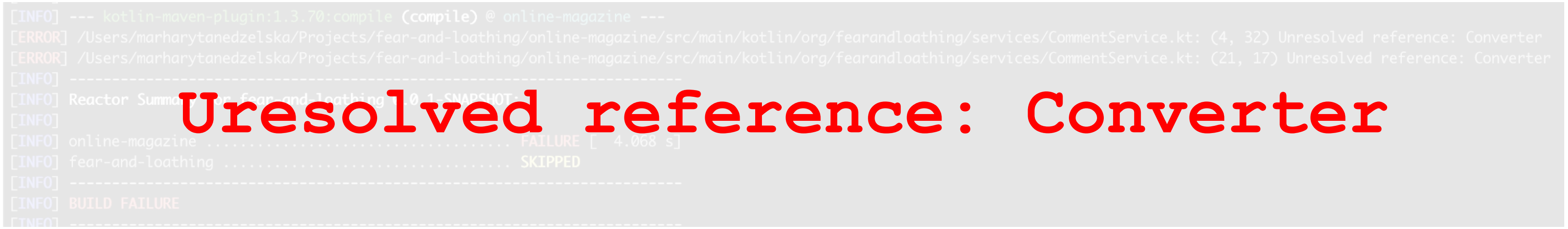
```
import org.fearandloathing.dto.Comment //Scala class

interface CommentService {

   fun getComment(id: Long): Comment

}

@Service class CommentServiceImpl(@Autowired private val

            commentRepository: CommentRepository): CommentService {

   override fun getComment(id: Long) = {...}

                              …

}
```

OOOOOPS!



```
[INFO] --- kotlin-maven-plugin:1.3.70:compile (compile) @ online-magazine ---
[ERROR] /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/src/main/kotlin/org/fearandloathing/services/CommentService.kt: (4, 32) Unresolved reference: Converter
[ERROR] /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/src/main/kotlin/org/fearandloathing/services/CommentService.kt: (21, 17) Unresolved reference: Converter
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for fear-and-loathing 0.0.1-SNAPSHOT:
[INFO]
[INFO] online-magazine .................................... FAILURE [  4.068 s]
[INFO] fear-and-loathing .................................. SKIPPED
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
```

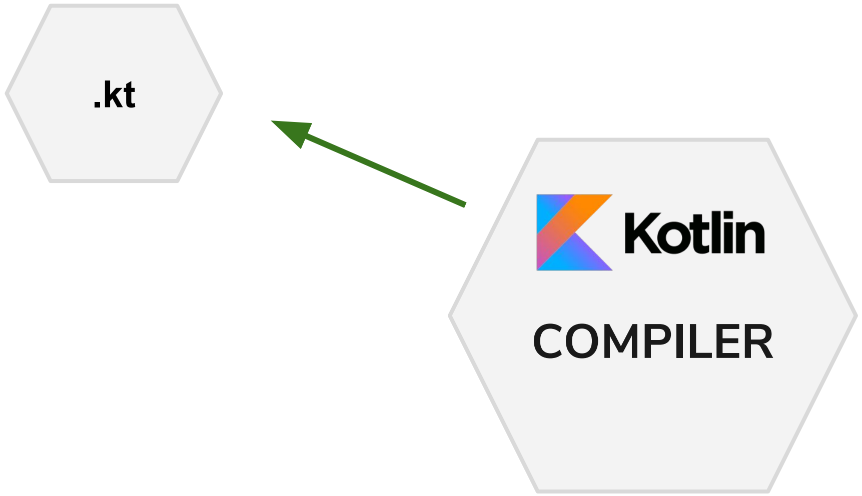
OOOOOPS!



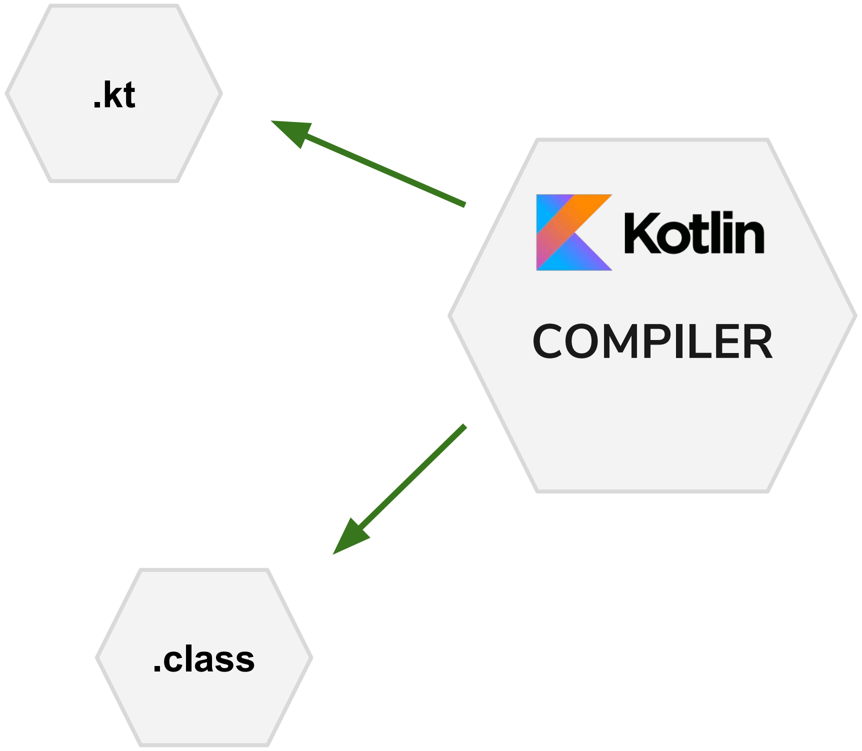
```
[INFO] --- kotlin-maven-plugin:1.3.70:compile (compile) @ online-magazine ---
[ERROR] /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/src/main/kotlin/org/fearandloathing/services/CommentService.kt: (4, 32) Unresolved reference: Converter
[ERROR] /Users/marharytanedzelska/Projects/fear-and-loathing/online-magazine/src/main/kotlin/org/fearandloathing/services/CommentService.kt: (21, 17) Unresolved reference: Converter
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for fear-and-loathing 0.0.1-SNAPSHOT:
[INFO]
[INFO] online-magazine .................................... FAILURE [  4.068 s]
[INFO] fear-and-loathing .................................. SKIPPED
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
```

Uresolved reference: Converter

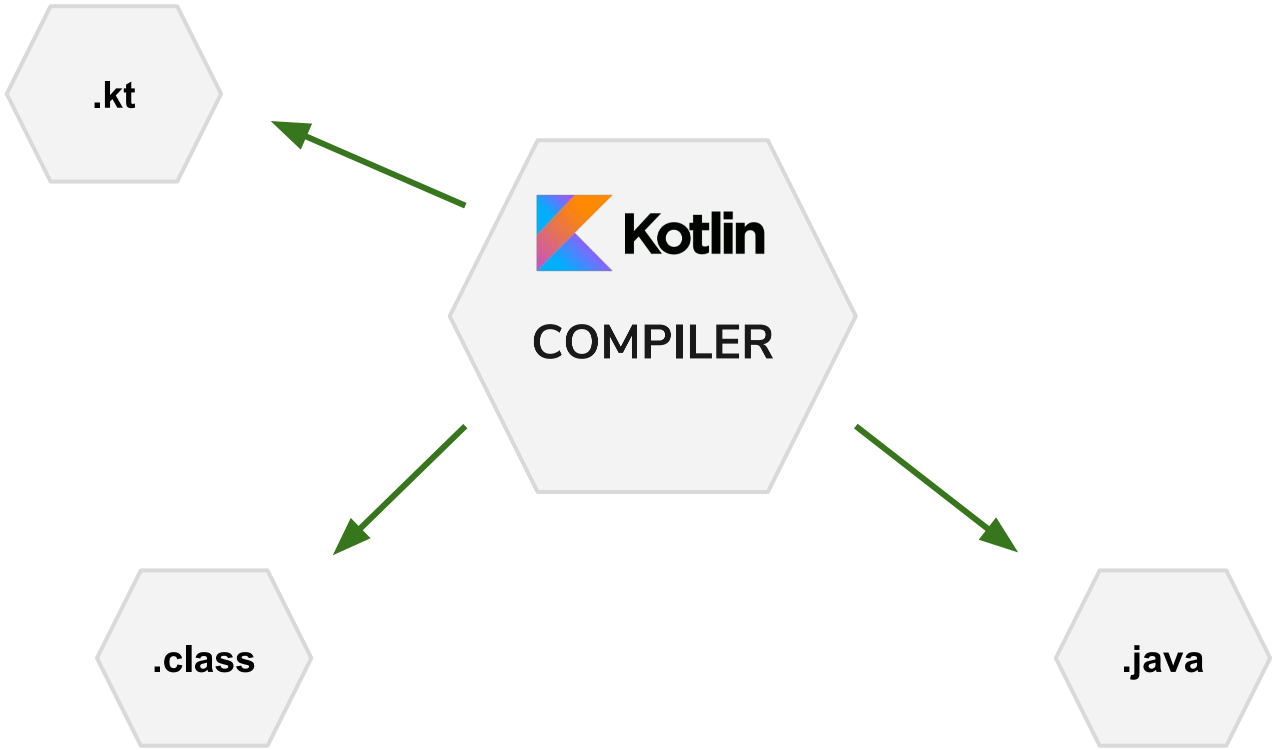
THE REVENGE OF A COMPILER



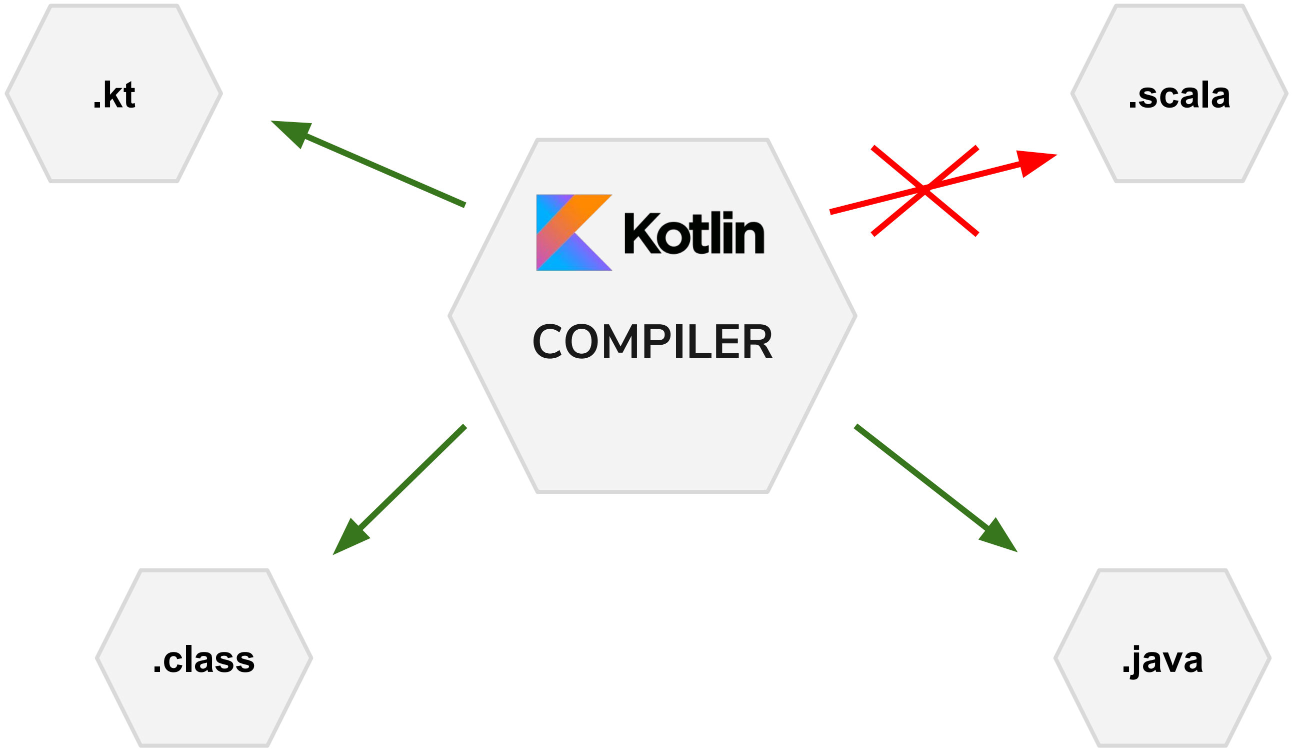
THE REVENGE OF A COMPILER



THE REVENGE OF A COMPILER



THE REVENGE OF A COMPILER



THE REVENGE OF A COMPILER

HERE COMES MODULARIZATION

SPLIT INTO MODULES

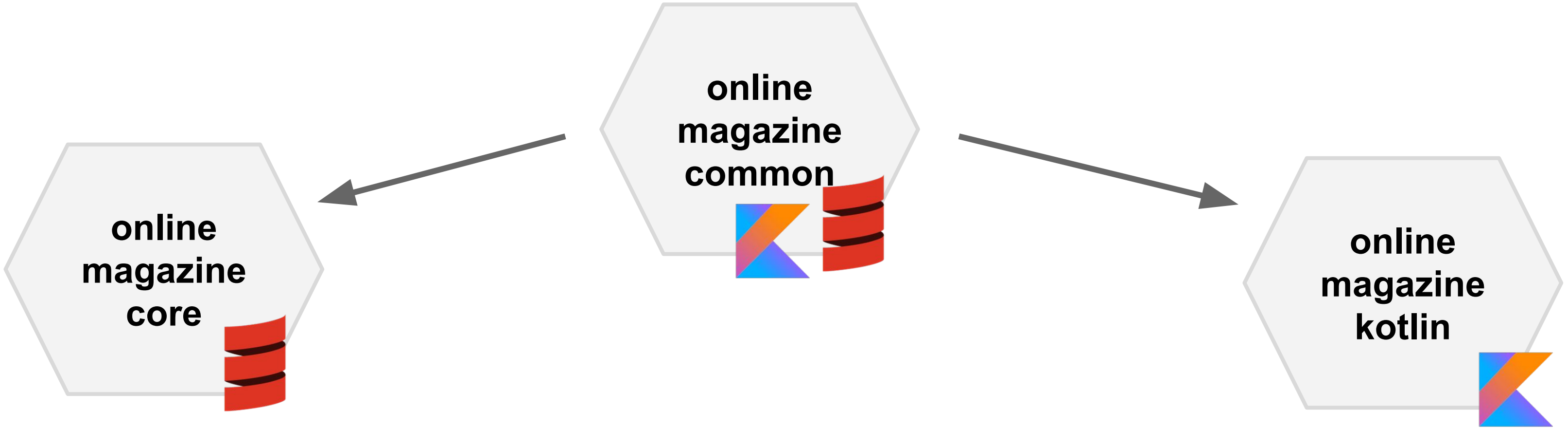
online
magazine
core

SPLIT INTO MODULES

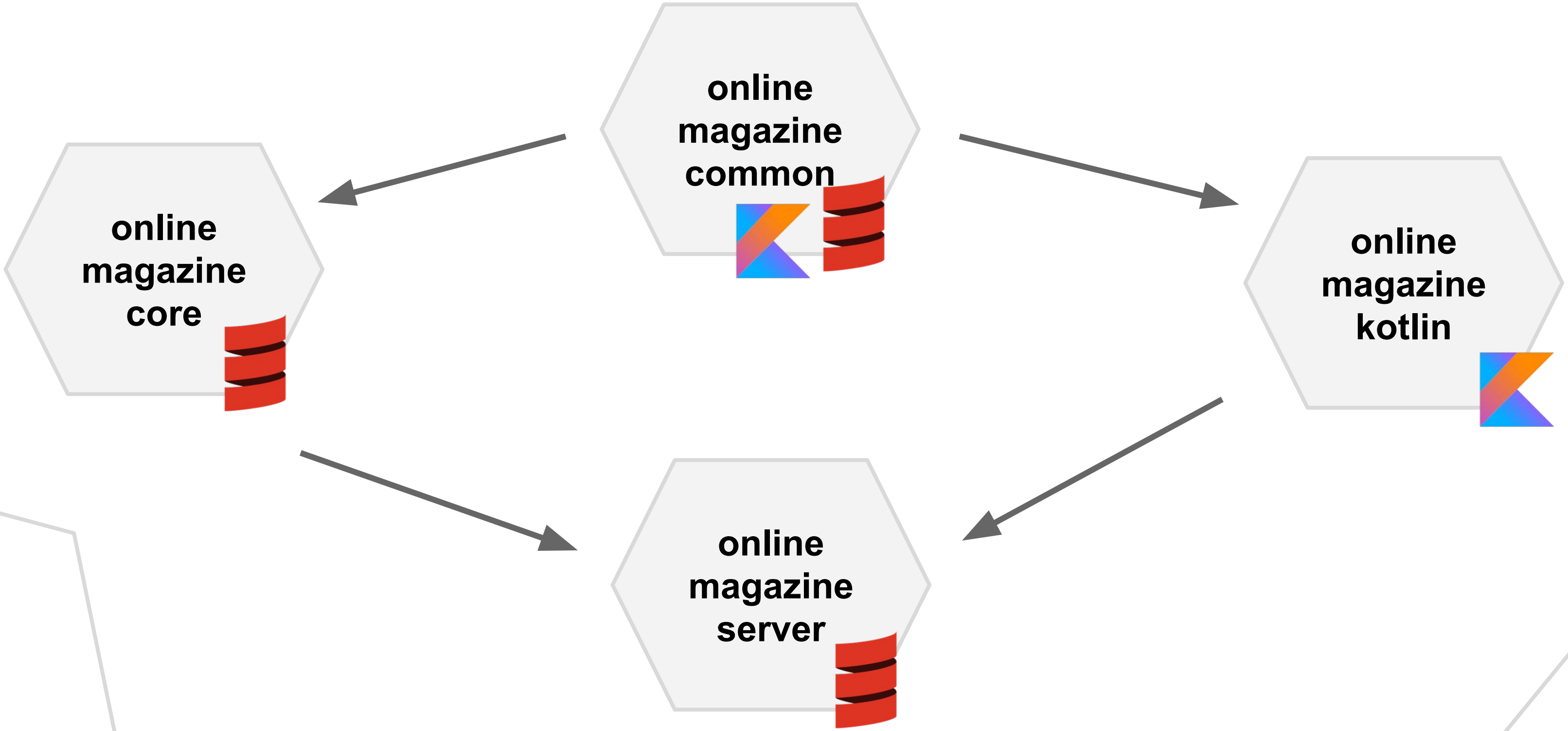
online
magazine
core

online
magazine
kotlin

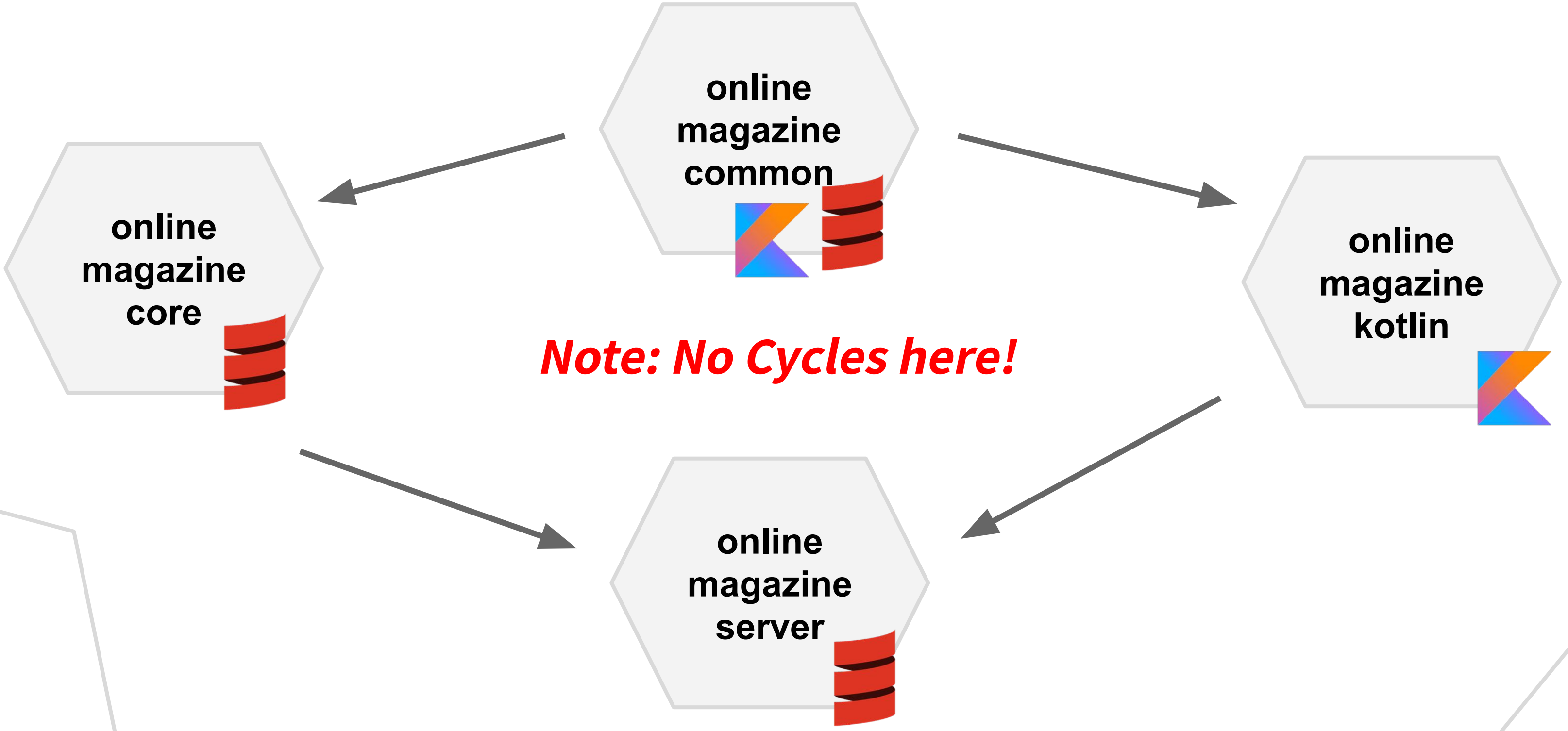
SPLIT INTO MODULES



SPLIT INTO MODULES



SPLIT INTO MODULES

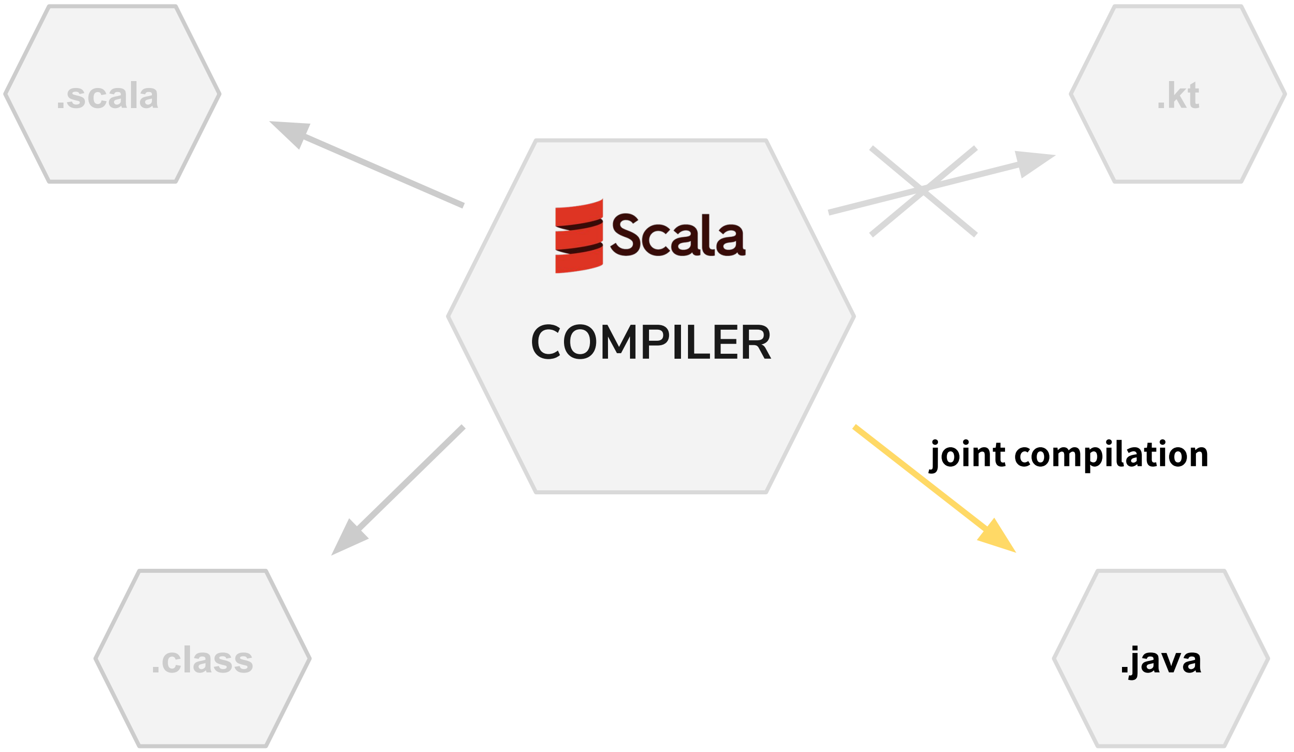


Note: No Cycles here!

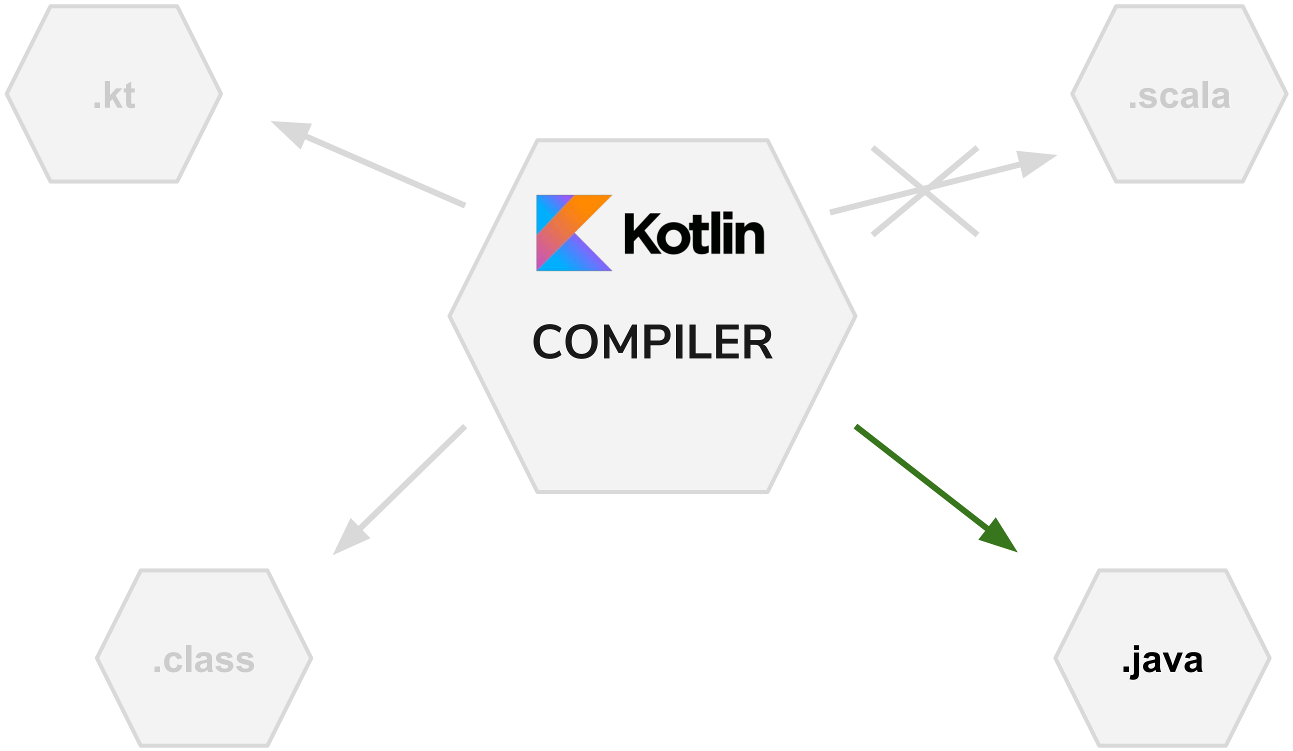
THE REVENGE OF A COMPILER

WHAT ABOUT JAVA?

THE REVENGE OF A COMPILER

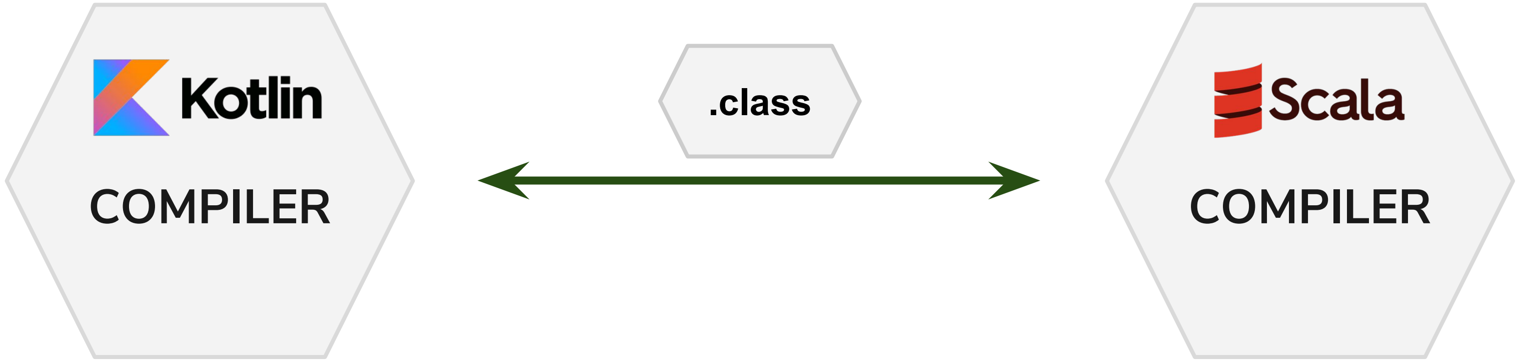


THE REVENGE OF A COMPILER



ATTENTION!
A BIG SPOILER!

SPOILER!



PROBLEM 2

SCALA COLLECTIONS

once you get locked into a serious drug collection,
the tendency is to push it as far as you can

SCALA COLLECTIONS

Scala collections $\neq$ **Java collections**

SCALA COLLECTIONS

Scala collections ≠ Java collections

Kotlin collections = Java collections

SCALA COLLECTIONS

Kotlin collections $\neq$ **Scala collections**

SCALA COLLECTION TO KOTLIN

```
val scalaList = /* some code here */
```

SCALA COLLECTION TO KOTLIN

```
import scala.jdk.CollectionConverters._


val scalaList = /* some code here */

val kotlinList = asJavaListConverter(scalaList).asJava()
```

SCALA COLLECTIONS IN KOTLIN… WHY NOT?

SCALA COLLECTION TO KOTLIN

```
val scalaList = /* some code here */

val result =
    scalaList.map({...}).filter({...}).find({...})
```

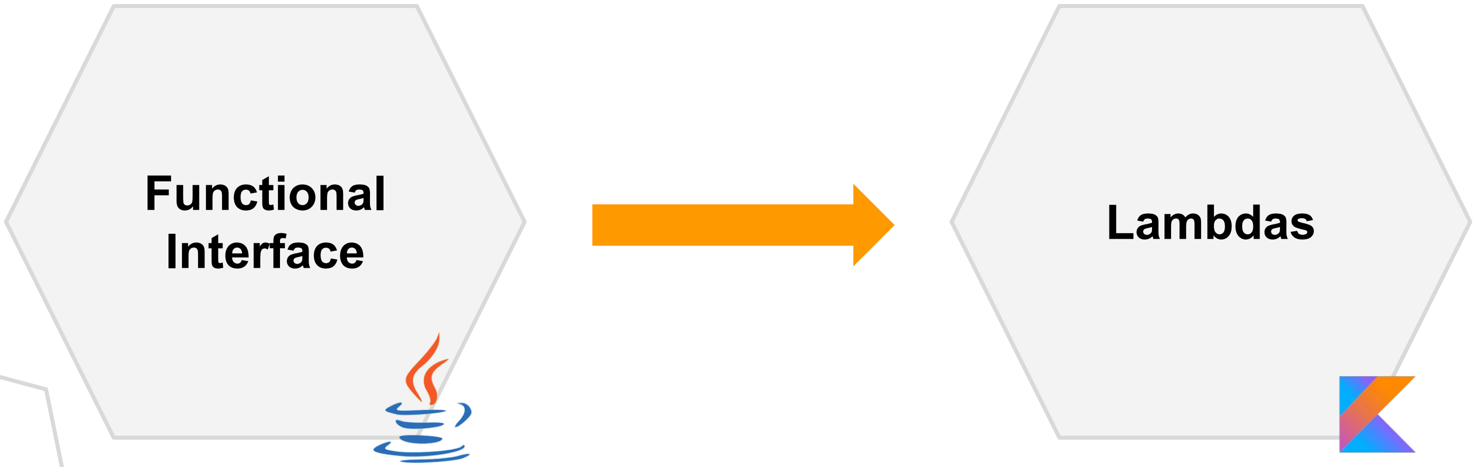
SCALA COLLECTIONS

LAMBDAS

SCALA COLLECTIONS

Functional Interface

SCALA COLLECTIONS



SCALA COLLECTIONS AS IS

```
val scalaList = /* some code here */

val kotlinList = scalaList.map { it.name }
```

ONE DOES NOT SIMPLY
USE KOTLIN LAMBDAS IN SCALA COLLECTIONS
gflip.com

ONE DOES NOT SIMPLY
USE KOTLIN LAMBDAS IN SCALA COLLECTIONS
ngflip.com

SCALA COLLECTIONS AS IS

```
public interface Function1 {

  Object apply(final Object v1);

  Function1 compose(final Function1 g);

  Function1 andThen(final Function1 g);

}
```

SCALA COLLECTIONS AS IS

```
val scalaList = /* some code here */

val kotlinList = scalaList.map (object:
    AbstractFunction1<Article, Any>() {
        override fun apply(v1: Article): Any =
            v1.title == title
    }
)
```

SCALA COLLECTIONS

WAIT!!!
SCALA 2.13

SCALA COLLECTIONS AS IS

```
public interface Function1 {

  Object apply(final Object v1);

  // $FF: synthetic method
  static Function1 compose$(final
Function1 $this, final Function1 g) {
      return $this.compose(g);
  }

  default Function1 compose(final
Function1 g) {
      return (x) -> {
         return this.apply(g.apply(x));
      };
  }

  // $FF: synthetic method
  static Function1 andThen$(final
Function1 $this, final Function1 g) {
      return $this.andThen(g);
  }

  default Function1 andThen(final
Function1 g) {
      return (x) -> {
         return g.apply(this.apply(x));
      };
  }
}
```

SCALA COLLECTIONS AS IS

```
public interface Function1 {

  Object apply(final Object v1);

  // $FF: synthetic method
  static Function1 compose$(final
Function1 $this, final Function1 g) {
      return $this.compose(g);
  }

  default Function1 compose(final
Function1 g) {
      return (x) -> {
         return this.apply(g.apply(x));
      };
  }
```

```
  // $FF: synthetic method
  static Function1 andThen$(final
Function1 $this, final Function1 g) {
      return $this.andThen(g);
  }

  default Function1 andThen(final
Function1 g) {
      return (x) -> {
         return g.apply(this.apply(x));
      };
  }
}
```

SCALA COLLECTIONS AS IS

```
val scalaList = /* some code here */

val kotlinList = scalaList.map { it.name }
```

SCALA COLLECTIONS AS IS

```
val scalaList = /* some code here */

val kotlinList = scalaList.map { it.name }
```

LET’S LOOK AT POLL RESULTS!

PROBLEM 3

SCALA IMPLICITS

IMPLICIT EXAMPLE

```
class Test {



    def f(a: Int) (implicit b: Int): Int = {
        a + b
    }




}
```

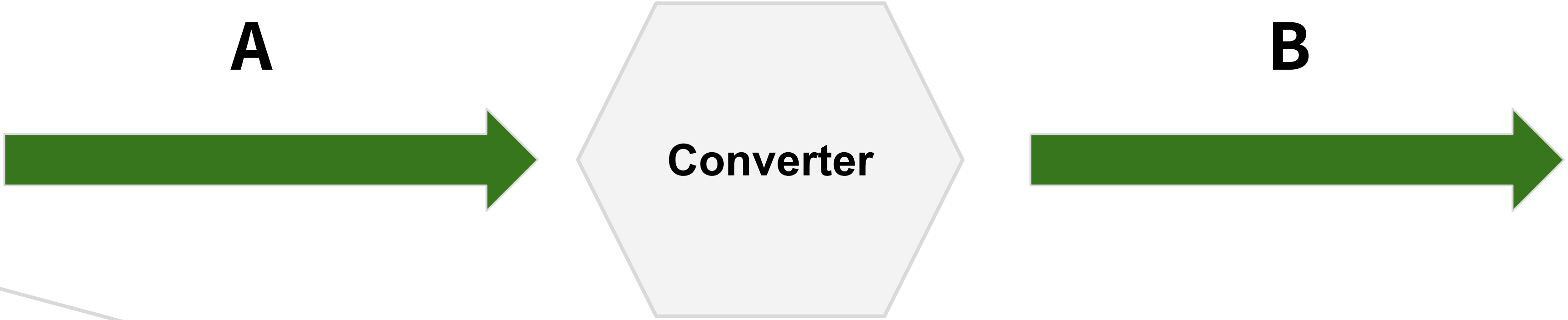
IMPLICIT EXAMPLE

```
class Test {

    private implicit val B: Int = 5

    def f(a: Int) (implicit b: Int): Int = {
        a + b
    }




}
```

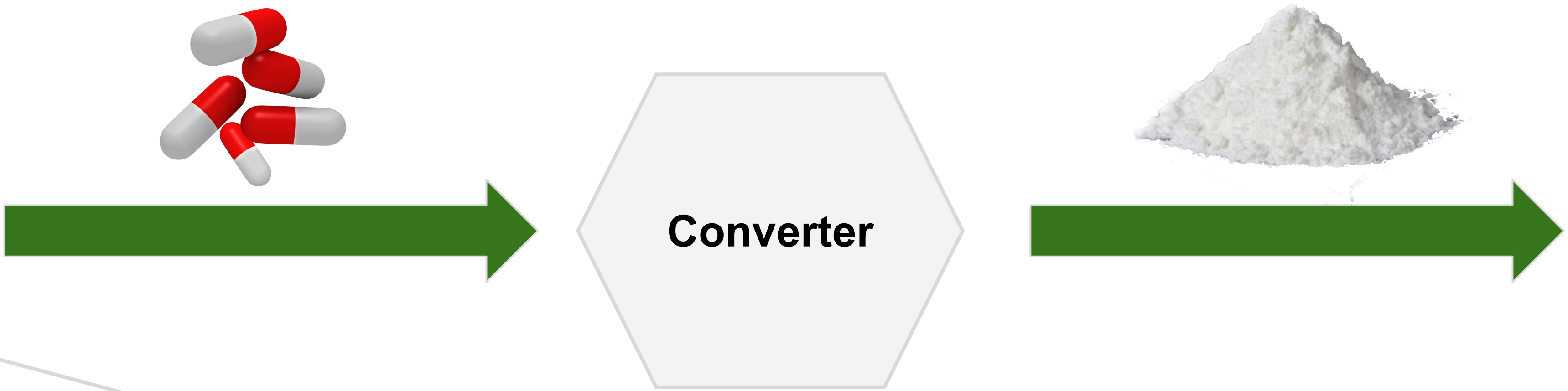
IMPLICIT EXAMPLE

```
class Test {

    private implicit val B: Int = 5

    def f(a: Int) (implicit b: Int): Int = {
        a + b
    }

    fun main(args: Array[String]): Unit = {
       new Test().f(2) // 2 + 5 = 7
    }
}
```

SCALA IMPLICITS EXAMPLE



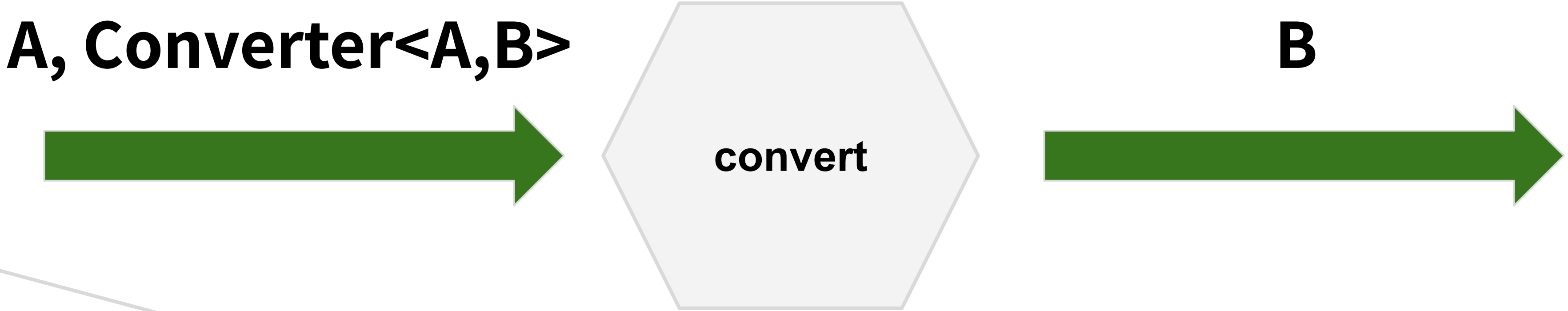
SCALA IMPLICITS EXAMPLE



IMPLICIT EXAMPLE

```
trait Converter[A,B] {
    def convert(a: A): B
}
```

SCALA IMPLICITS EXAMPLE



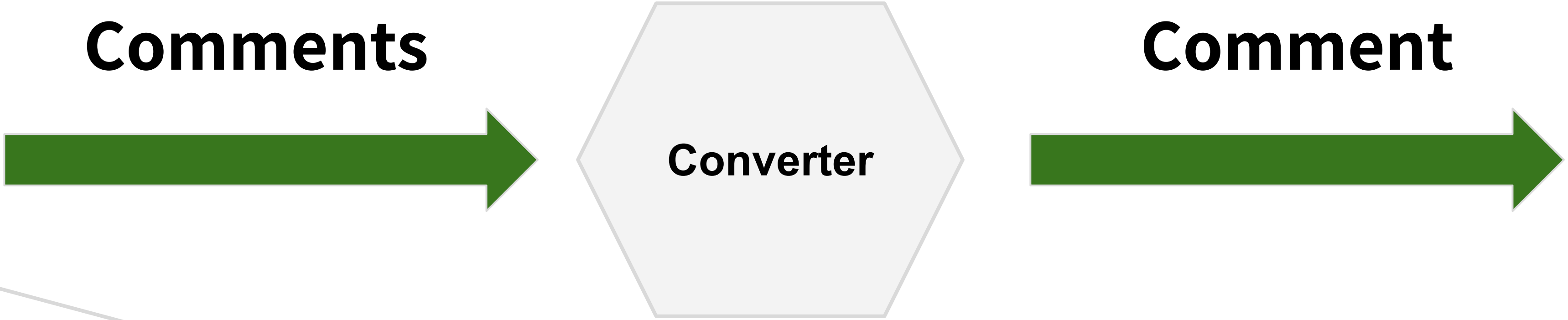
SCALA IMPLICITS EXAMPLE

FIRSTLY, WITHOUT IMPLICITS

IMPLICIT EXAMPLE

```scala
trait Converter[A,B] {
    def convert(a: A): B
}

object Converter {
    def convert[A,B](a: A, c: Converter[A,B]): B =
                                              c.convert(a)
    val convertCommentEntity: Converter[Comments,Comment]
        = (c: Comments) =>
    Comment(c.getId, c.getArticle, c.getBody, c.getAuthor)
 }
```

SCALA IMPLICITS EXAMPLE



IMPLICIT EXAMPLE

```
trait Converter[A,B] {
    def convert(a: A): B
}

object Converter {
    def convert[A,B](a: A, c: Converter[A,B]): B =
                                                     c.convert(a)
    val convertCommentEntity: Converter[Comments, Comment]
        = (c: Comments) =>
     Comment(c.getId, c.getArticle, c.getBody, c.getAuthor)
 }
```

IMPLICIT EXAMPLE

```
import org.fearandloathing.dto.Converter._


val entity: Comments = Comments(/* some code here */ )

val dto: Comment = convert(entity, convertCommentEntity)
```

SCALA IMPLICITS EXAMPLE

AND NOW WITH IMPLICITS

IMPLICIT EXAMPLE

```
trait Converter[A,B] {
    def convert(a: A): B
}
object Converter {
    def convert[A,B](a: A)(implicit c: Converter[A,B]):B =
                                                c.convert(a)
    implicit val convertCommentEntity:
                                Converter[Comments,Comment]=
        (c: Comments) => Comment(c.getId, c.getArticle,
                                      c.getBody, c.getAuthor)
 }
```

IMPLICIT EXAMPLE

```
trait Converter[A,B] {
    def convert(a: A): B
}
object Converter {
    def convert[A,B](a: A)(implicit c: Converter[A,B]):B =
                                                c.convert(a)
    implicit val convertCommentEntity:
                                Converter[Comments, Comment]=
        (c: Comments) => Comment(c.getId, c.getArticle,
                                      c.getBody, c.getAuthor)

 }
```

IMPLICIT EXAMPLE

```
trait Converter[A,B] {
    def convert(a: A): B
}
object Converter {
    def convert[A,B](a: A)(implicit c: Converter[A,B]):B =
                                                    c.convert(a)
    implicit val convertCommentEntity:
                                  Converter[Comments, Comment]=
        (c: Comments) => Comment(c.getId, c.getArticle,
                                          c.getBody, c.getAuthor)
 }
```

HOW

TO USE

IT?

IMPLICIT EXAMPLE

```
import org.fearandloathing.dto.Converter._


val entity: Comments = Comments(/* some code here */ )

val dto: Comment = convert(entity)
```

IMPLICIT EXAMPLE

HOW TO USE IT IN KOTLIN?

DECOMPILE

IMPLICIT EXAMPLE

DECOMPILE SCALA TO JAVA

IMPLICIT EXAMPLE (KOTLIN)

```
//decompiled from Converter.class
package org.fearandloathing.dto;

import scala.reflect.ScalaSignature;

@ScalaSignature(
  bytes = "..."
)
public interface Converter {
  static Converter convertCommentEntity() {
     return Converter$.MODULE$.convertCommentEntity();
  }
  Object convert( final Object a);
}

 //decompiled from Converter$.class
package org.fearandloathing.dto;

import org.fearandloathing.entity.Articles;
import org.fearandloathing.entity.Comments;
import org.fearandloathing.entity.Users;
import scala.Predef.;

public final class Converter$ {
  public static final Converter$ MODULE$ =  new Converter$();
  private static final Converter convertCommentEntity =  new Converter() {
     public Comment convert( final Comments c) {
        return new Comment(.MODULE$.Long2long(c.getId()), .MODULE$.Long2long(c.getArticle()),
c.getBody(), .MODULE$.Long2long(c.getAuthor()));
     }
     // $FF: synthetic method
     // $FF: bridge method
     public Object convert( final Object a) {
        return this.convert((Comments)a);
     }
  };

  public Object convert( final Object a,  final Converter c) {
     return c.convert(a);
  }

  public Converter convertCommentEntity() {
     return convertCommentEntity;
  }

  private Converter$() {
  }
}
```

IMPLICIT EXAMPLE (KOTLIN)

```
public final class Converter$ {
    public static final Converter$ MODULE$ = new Converter$();
    private static final Converter convertCommentEntity = new
Converter() {...};

    public Object convert(final Object a, final Converter c) {
            return c.convert(a);
    }

    public Converter convertCommentEntity() {
            return convertCommentEntity;
    }

    private Converter$() {
    }
}
```

IMPLICIT EXAMPLE (KOTLIN)

```
public final class Converter$ {
    public static final Converter$ MODULE$ =  new Converter$();
    private static final Converter convertCommentEntity =  new
Converter() {...};

    public Object convert(final Object a, final Converter c) {
            return c.convert(a);
    }

    public Converter convertCommentEntity() {
            return convertCommentEntity;
    }

    private Converter$() {
    }
}
```

IMPLICIT EXAMPLE (KOTLIN)

```
public final class Converter$ {
    public static final Converter$ MODULE$ = new Converter$();
    private static final Converter convertCommentEntity = new
Converter() {...};

    public Object convert(final Object a, final Converter c) {
            return c.convert(a);
    }

    public Converter convertCommentEntity() {
            return convertCommentEntity;
    }

    private Converter$() {
    }
}
```

IMPLICIT EXAMPLE (KOTLIN)

```
public final class Converter$ {
    public static final Converter$ MODULE$ =  new Converter$();
    private static final Converter convertCommentEntity =  new
Converter() {...};

    public Object convert(final Object a, final Converter c) {
            return c.convert(a);
    }

    public Converter convertCommentEntity() {
            return convertCommentEntity;
    }

    private Converter$() {
    }
}
```

IMPLICIT EXAMPLE (KOTLIN)

```
import org.fearandloathing.dto.`Converter$`.`MODULE$` as
                                                        converter


val entity: Comments = Comments(/* some code here */ )

val dto: Comment = converter.convert(entity, ...)
```

IMPLICIT EXAMPLE (KOTLIN)

```
public final class Converter$ {
    public static final Converter$ MODULE$ =  new Converter$();
    private static final Converter convertCommentEntity =  new
Converter() {
        public Comment convert(final Comments c) {
                return new Comment(.MODULE$.Long2long(c.getId()),
                .MODULE$.Long2long(c.getArticle()), c.getBody(),
            .MODULE$.Long2long(c.getAuthor()));
            }
                                        ...
    };
                                        ...

    public Converter convertCommentEntity() {
            return convertCommentEntity;
    }

}
```

IMPLICIT EXAMPLE (KOTLIN)

```
public final class Converter$ {
    public static final Converter$ MODULE$ = new Converter$();
    private static final Converter convertCommentEntity = new
Converter() {
        public Comment convert(final Comments c) {
                return new Comment(.MODULE$.Long2long(c.getId()),
                .MODULE$.Long2long(c.getArticle()), c.getBody(),
            .MODULE$.Long2long(c.getAuthor()));
            }
                                        ...
    };
                                        ...

    public Converter convertCommentEntity() {
            return convertCommentEntity;
    }

}
```

IMPLICIT EXAMPLE (KOTLIN)

```
import org.fearandloathing.dto.`Converter$`.`MODULE$` as
                                                  converter


val entity: Comments = Comments(/* some code here */ )

val dto: Comment = converter.convert(entity,
      converter.convertCommentEntity())
```

IMPLICIT EXAMPLE (KOTLIN)

```
//decompiled from Converter.class
package org.fearandloathing.dto;

import scala.reflect.ScalaSignature;

@ScalaSignature(
  bytes = "..."
)
public interface Converter {
	static Converter convertCommentEntity() {
		return Converter$.MODULE$.convertCommentEntity();
	}
	Object convert(final Object a);
}
```

IMPLICIT EXAMPLE (KOTLIN)

```
import org.fearandloathing.dto.Converter


val entity: Comments = Comments(/* some code here */ )

val dto: Comment =
      Converter.convertCommentEntity().convert(entity)
```

IMPLICIT EXAMPLE

WHAT ABOUT JAVA?

IMPLICIT EXAMPLE

```
import org.fearandloathing.dto.Converter;


Comments entity = new Comments(/* some code here */ );

Comment dto = Converter.convertCommentEntity()
                    .convert(entity);
```

IMPLICIT EXAMPLE

```
import org.fearandloathing.dto.Converter;


val entity = new Comments(/* some code here */ );

val dto = Converter.convertCommentEntity()
                     .convert(entity);
```

PROBLEM 4

OPERATORS OVERLOADING

IN SCALA YOU CAN OVERLOAD EVERYTHING

OPERATORS OVERLOADING

!##W$%^%$*^%#$%#%$^%E^F

^%$&&^%&%$*^%*&%R%%^%^T

^RF%$%$^%*%^$%&^%$^%E@#

@%$$%@%@$#@!$%^&*(*&^%$

%^&*()%^$$%^&%$#$%

OPERATORS OVERLOADING

```scala
case class Comment(@BeanProperty id: Long,
                  @BeanProperty article: Long,
                  @BeanProperty body: String,
                  @BeanProperty author: Long) {

   def ~~~ (author: Long): Comment =
       this.copy(author = author)

}
```

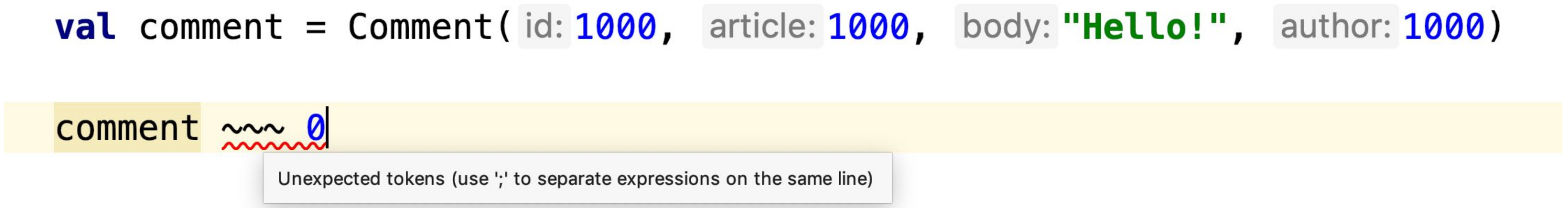
OPERATORS OVERLOADING

```
case class Comment(@BeanProperty id: Long,
                   @BeanProperty article: Long,
                   @BeanProperty body: String,
                   @BeanProperty author: Long) {

  def ~~~ (author: Long): Comment =
    this.copy(author = author)

  def f() = {
    this ~~~ 0
    this.~~~(0)
  }
}
```

OPERATORS OVERLOADING

HOW TO CALL IT FROM KOTLIN

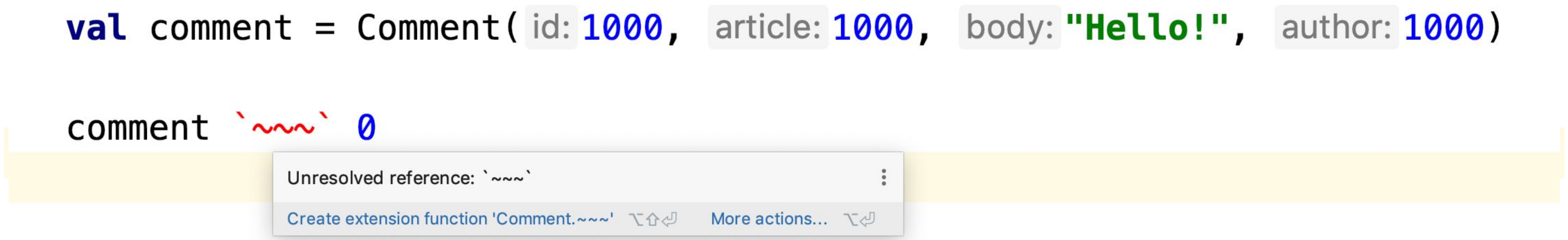
OPERATORS OVERLOADING



```
val comment = Comment( id: 1000,  article: 1000,  body: "Hello!",  author: 1000)

comment ~~~ 0
```

Unexpected tokens (use ';' to separate expressions on the same line)

OPERATORS OVERLOADING

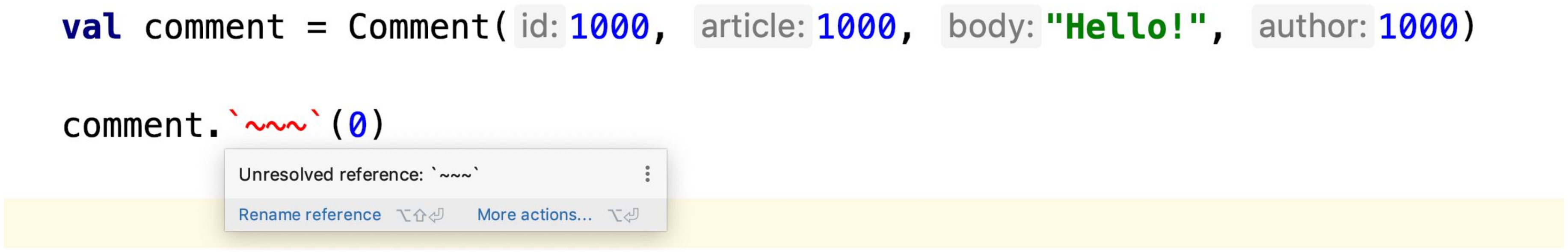


```
val comment = Comment( id: 1000,  article: 1000,  body: "Hello!",  author: 1000)

comment `~~~` 0
```

Unresolved reference: `~~~`

Create extension function 'Comment.~~~' ⌥⇧⏎   More actions... ⌥⏎

OPERATORS OVERLOADING



```
val comment = Comment( id: 1000, article: 1000, body: "Hello!", author: 1000)

comment.`~~~`(0)
```

Unresolved reference: `~~~`

DECOMPILE

IMPLICIT EXAMPLE

DECOMPILE SCALA TO JAVA

OPERATORS OVERLOADING

```
public class Comment implements Product, Serializable {

                                        ...

    public Comment $tilde$tilde$tilde(final long author) {
                    ...
    }

                                        ...

}
```

OPERATORS OVERLOADING

```
import org.fearandloathing.dto.Comment


val comment = Comment(/*Some code here*/)
val copiedComment = comment.`$tilde$tilde$tilde`(userId)
```

OPERATORS OVERLOADING

WHAT ABOUT JAVA?

OPERATORS OVERLOADING

```
import org.fearandloathing.dto.Comment


final var comment = new Comment(1000, 1000, "Hello from
                                                  Java!", 1);

comment.$tilde$tilde$tilde(0);
```

Too weird to live, too rare to die

OPERATORS OVERLOADING

ONE MORE INTERESTING THING

OPERATORS OVERLOADING

```
case class Comment(@BeanProperty id: Long,
                   @BeanProperty article: Long,
                   @BeanProperty body: String,
                   @BeanProperty author: Long) {


    def plus(comment: Comment): Comment =
        Comment(id, article, body + comment.body, author)


}
```

OPERATORS OVERLOADING

HOW TO USE IT IN KOTLIN?

OPERATORS OVERLOADING

```
import org.fearandloathing.dto.Comment

val c1 = Comment(1,1, "A", 1)
val c2 = Comment(2,1, "B", 1)

c1.plus(c2)
```

OPERATORS OVERLOADING

```kotlin
import org.fearandloathing.dto.Comment

val c1 = Comment(1,1, "A", 1)
val c2 = Comment(2,1, "B", 1)

c1.plus(c2)

c1 + c2
```

OPERATORS OVERLOADING

KOTLIN OPERATORS

OPERATORS OVERLOADING

```
class Comments: Serializable {
                                             …

    operator fun invoke(newAuthor: Long): Comments {
        val entity = Comments()
                entity.article = this.article
                entity.body = this.body
                entity.author = newAuthor
                return entity
    }
}
```

OPERATORS OVERLOADING

```kotlin
val entity = Comments()

val plagiarism = entity(1)

println(plagiarism)
```

OPERATORS OVERLOADING

```
val entity: Comments = new Comments()

val plagiarism = entity.invoke(5)

println(plagiarism)
```

OPERATORS OVERLOADING

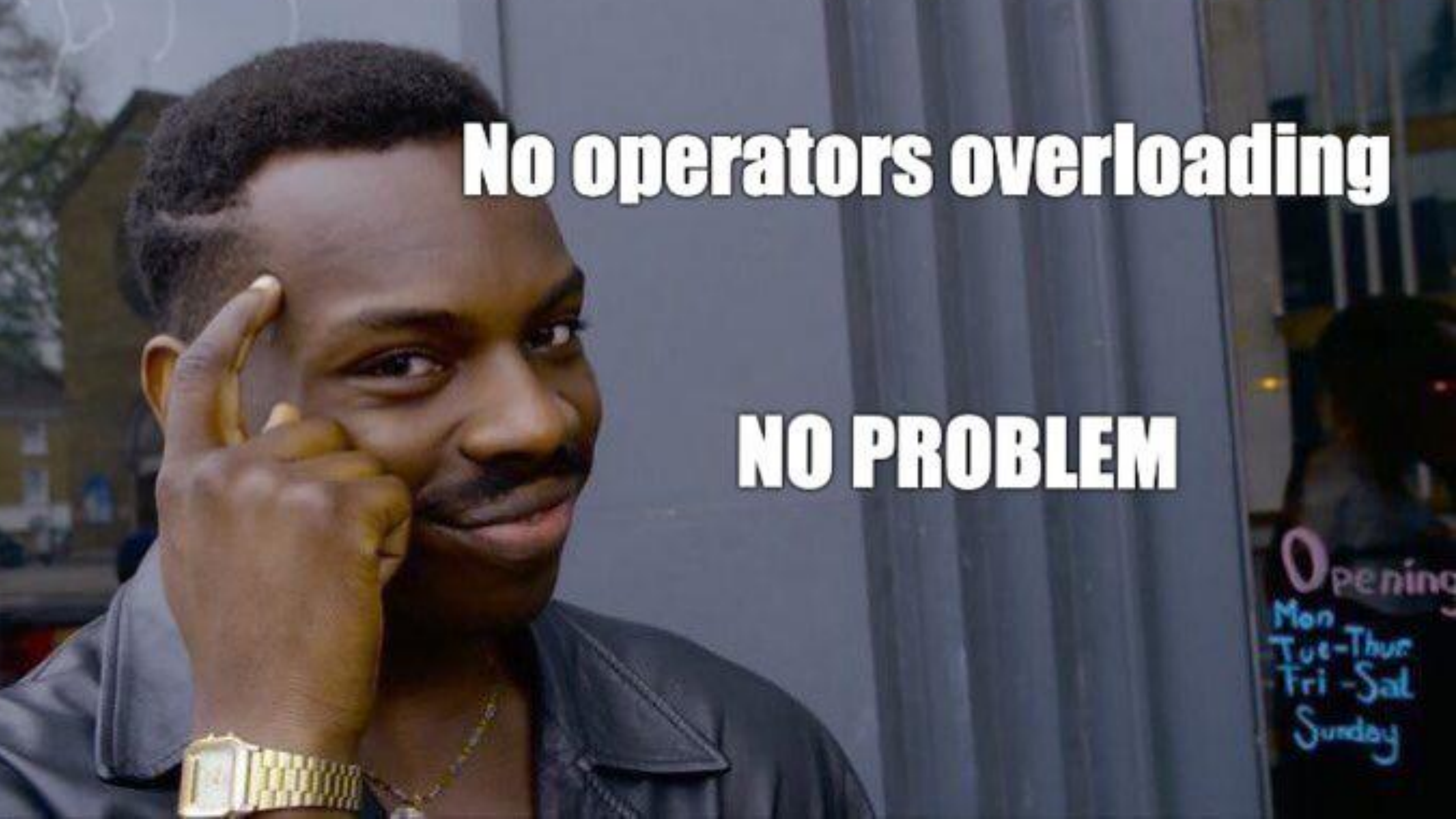
```
Comments entity = new Comments();

Comments println(plagiarism) = entity.invoke(10);

System.out.println(println(plagiarism));
```

OPERATORS OVERLOADING

```
final var entity = new Comments();

final var value = entity.invoke(10);

System.out.println(value);
```

OPERATORS OVERLOADING

JAVA OPERATORS


No operators overloading
NO PROBLEM
Opening
Mon
Tue-Thur
Fri -Sat
Sunday

PROBLEM 5

SCALA IDE PLUGIN

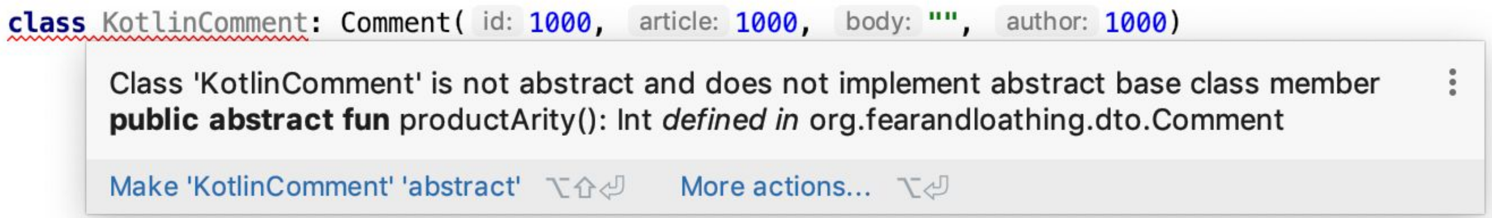
SCALA IDE PLUGIN

WHERE IS THE PROBLEM?

SCALA IDE PLUGIN

Compiler VS

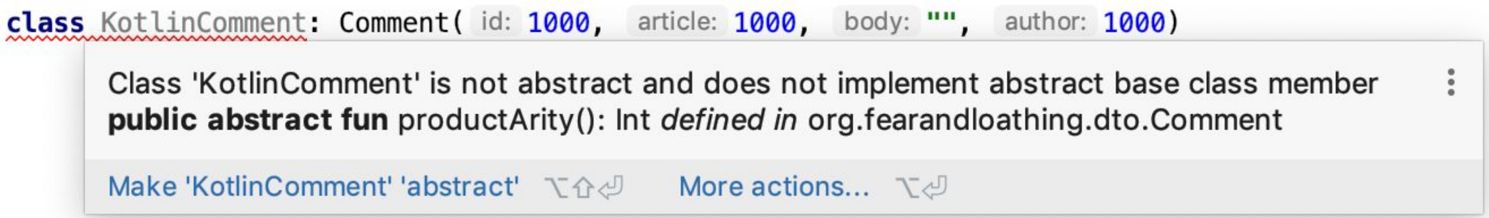
CASE CLASS INHERITANCE



```
class KotlinComment: Comment( id: 1000, article: 1000, body: "", author: 1000)
```

Class 'KotlinComment' is not abstract and does not implement abstract base class member **public abstract fun** productArity(): Int *defined in* org.fearandloathing.dto.Comment

CASE CLASS INHERITANCE



```
class KotlinComment: Comment( id: 1000, article: 1000, body: "", author: 1000)
```

Class 'KotlinComment' is not abstract and does not implement abstract base class member **public abstract fun** productArity(): Int *defined in* org.fearandloathing.dto.Comment

```
case class Comment(@BeanProperty id: Long,
                  @BeanProperty article: Long,
                  @BeanProperty body: String,
                  @BeanProperty author: Long)
```

YOU
SHALL NOT PASS

COMPILER SAYS

COMPILER SAYS

TYPE BOUNDS

```
class CommentContainer[C <: Comment](c: C) {
    def comment: C = c
}
```

BOUNDED TYPES

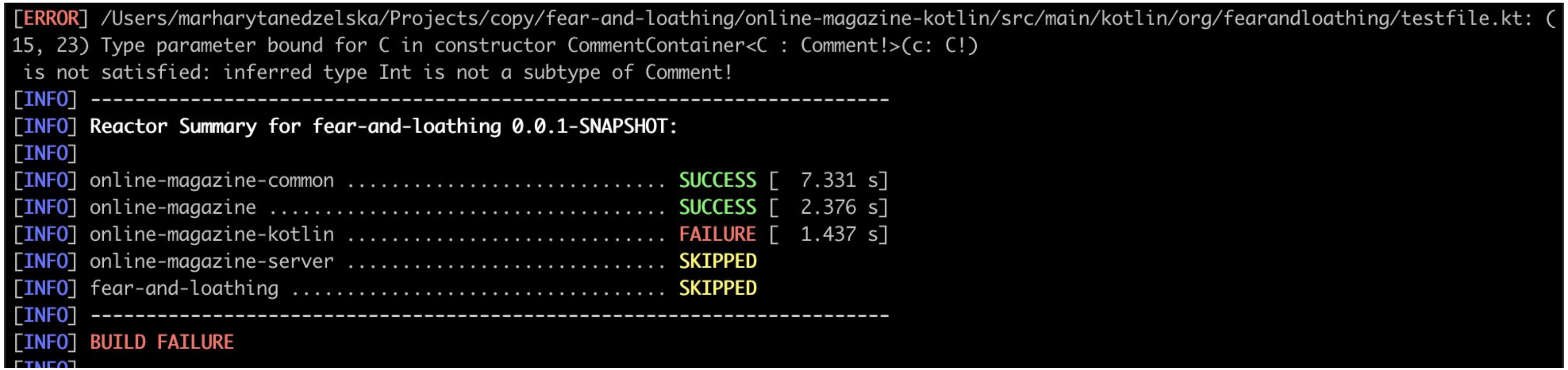
```scala
class CommentContainer[C <: Comment](c: C) {
    def comment: C = c
}


val failedContainer = CommentContainer(123)
```

BOUNDED TYPES

```scala
class CommentContainer[C <: Comment](c: C) {
    def comment: C = c
}


val failedContainer = CommentContainer(123)
```

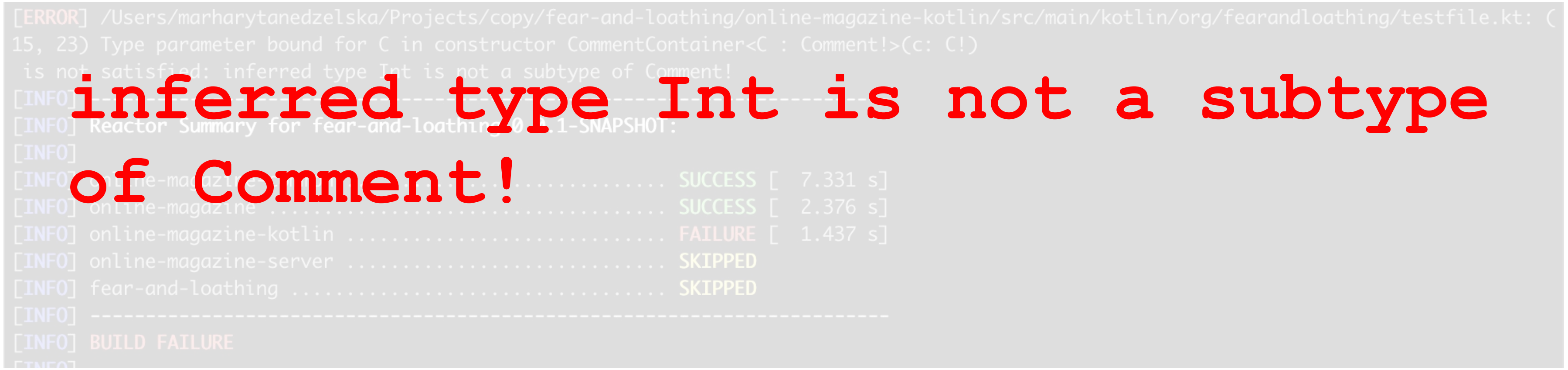
COMPILER SAYS

COMPILER SAYS



```
[ERROR] /Users/marharytanedzelska/Projects/copy/fear-and-loathing/online-magazine-kotlin/src/main/kotlin/org/fearandloathing/testfile.kt: (
15, 23) Type parameter bound for C in constructor CommentContainer<C : Comment!>(c: C!)
 is not satisfied: inferred type Int is not a subtype of Comment!
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for fear-and-loathing 0.0.1-SNAPSHOT:
[INFO]
[INFO] online-magazine-common ............................. SUCCESS [  7.331 s]
[INFO] online-magazine .................................... SUCCESS [  2.376 s]
[INFO] online-magazine-kotlin ............................. FAILURE [  1.437 s]
[INFO] online-magazine-server ............................. SKIPPED
[INFO] fear-and-loathing .................................. SKIPPED
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
```

COMPILER SAYS



```
[ERROR] /Users/marharytanedzelska/Projects/copy/fear-and-loathing/online-magazine-kotlin/src/main/kotlin/org/fearandloathing/testfile.kt: (
15, 23) Type parameter bound for C in constructor CommentContainer<C : Comment!>(c: C!)
 is not satisfied: inferred type Int is not a subtype of Comment!
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for fear-and-loathing 0.0.1-SNAPSHOT:
[INFO]
[INFO] online-magazine-common ............................ SUCCESS [  7.331 s]
[INFO] online-magazine ................................... SUCCESS [  2.376 s]
[INFO] online-magazine-kotlin ............................ FAILURE [  1.437 s]
[INFO] online-magazine-server ............................ SKIPPED
[INFO] fear-and-loathing ................................. SKIPPED
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
```

inferred type Int is not a subtype of Comment!

EVERYBODY LIES
EXCEPT FOR COMPILER
imgflip.com

PROBLEM 6

EXTENSION FUNCTIONS

YEAH, IF WE COULD ALL GET AN EXTENSION
THAT'D BE GREAT
makeameme.org

EXTENSION FUNCTIONS

```
val comments: Comments = new Comments()

val commentsCopy = comments.copy()

println(commentsCopy)
```

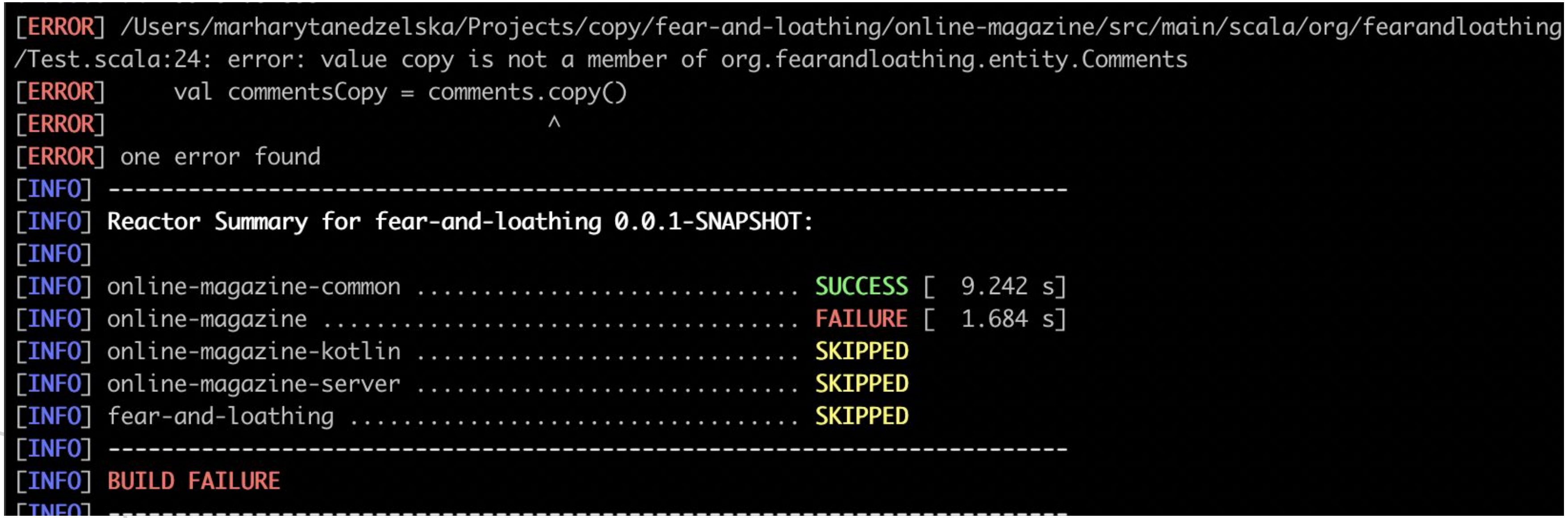
EXTENSION FUNCTIONS

```
fun Comments.copy(): Comments {

    val entity = Comments()

    entity.article = this.article
    entity.body = this.body
    entity.author = this.author

    return entity
}
```

EXTENSION FUNCTIONS

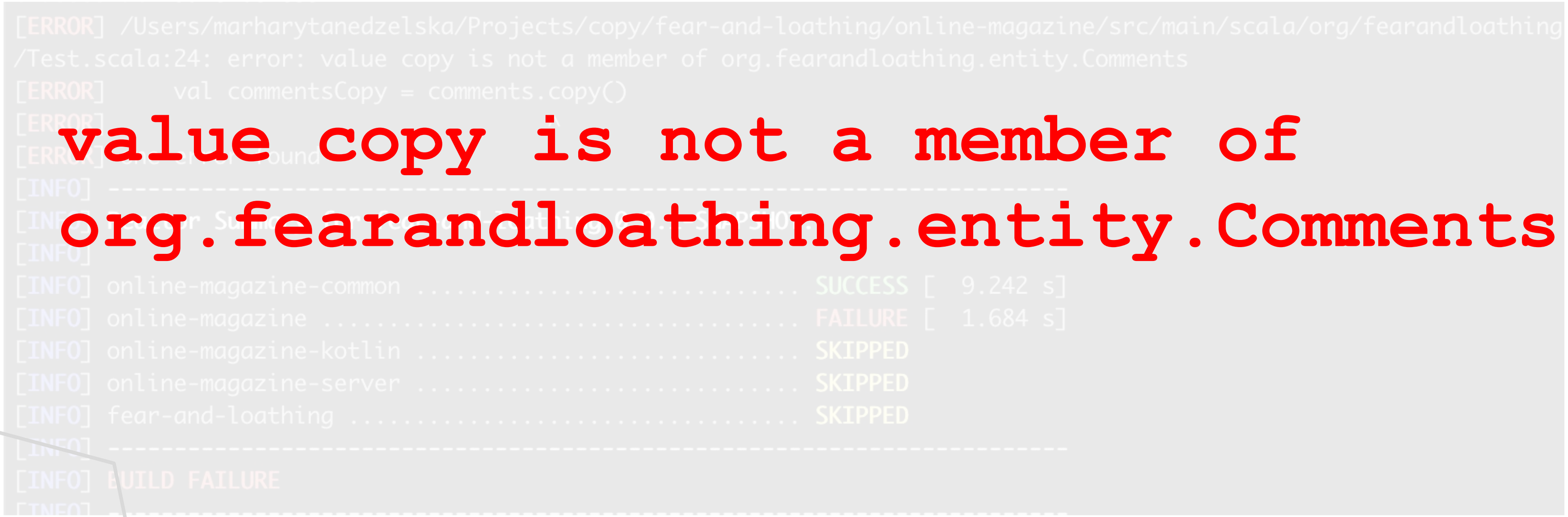
```
val comments: Comments = new Comments()

val commentsCopy = comments.copy()

println(commentsCopy)
```

EXTENSION FUNCTIONS



```
[ERROR] /Users/marharytanedzelska/Projects/copy/fear-and-loathing/online-magazine/src/main/scala/org/fearandloathing
/Test.scala:24: error: value copy is not a member of org.fearandloathing.entity.Comments
[ERROR]      val commentsCopy = comments.copy()
[ERROR]                                  ^
[ERROR] one error found
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for fear-and-loathing 0.0.1-SNAPSHOT:
[INFO]
[INFO] online-magazine-common ............................. SUCCESS [  9.242 s]
[INFO] online-magazine .................................... FAILURE [  1.684 s]
[INFO] online-magazine-kotlin ............................. SKIPPED
[INFO] online-magazine-server ............................. SKIPPED
[INFO] fear-and-loathing .................................. SKIPPED
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
```

EXTENSION FUNCTIONS



```
[ERROR] /Users/marharytanedzelska/Projects/copy/fear-and-loathing/online-magazine/src/main/scala/org/fearandloathing
/Test.scala:24: error: value copy is not a member of org.fearandloathing.entity.Comments
[ERROR]      val commentsCopy = comments.copy()
[ERROR]                                       ^
[ERROR] one error found
[INFO] ------------------------------------------------------------------------
[INFO] Reactor Summary for fear-and-loathing 0.0.1-SNAPSHOT:
[INFO]
[INFO] online-magazine-common ............................. SUCCESS [  9.242 s]
[INFO] online-magazine .................................... FAILURE [  1.684 s]
[INFO] online-magazine-kotlin ............................. SKIPPED
[INFO] online-magazine-server ............................. SKIPPED
[INFO] fear-and-loathing .................................. SKIPPED
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
```

value copy is not a member of org.fearandloathing.entity.Comments

DECOMPILE

EXTENSION FUNCTIONS

```
public final class CommentsKt {
   @NotNull
   public static final Comments copy(@NotNull Comments
$this$copy) {
         Intrinsics.checkParameterIsNotNull($this$copy,
"$this$copy");
         Comments entity = new Comments();
         entity.setArticle($this$copy.getArticle());
         entity.setBody($this$copy.getBody());
         entity.setAuthor($this$copy.getAuthor());
         return entity;
   }
}
```

EXTENSION FUNCTIONS

```
val comments: Comments = new Comments()

val commentsCopy = CommentsKt.copy(comments)

println(commentsCopy)
```

EXTENSION FUNCTIONS

LET’S MAKE IT MORE READABLE

EXTENSION FUNCTIONS

```
@file:JvmName("CommentsHelper")
package org.fearandloathing.entity

                          ...

fun Comments.copy(): Comments {

                          ...

}
```

EXTENSION FUNCTIONS

```
val comments: Comments = new Comments()

val commentsCopy = CommentsHelper.copy(comments)

println(commentsCopy)
```

PROBLEM 7

ASYNCHRONOUS CODE

ASYNCHRONOUS CODE

LET’S START FROM SCALA

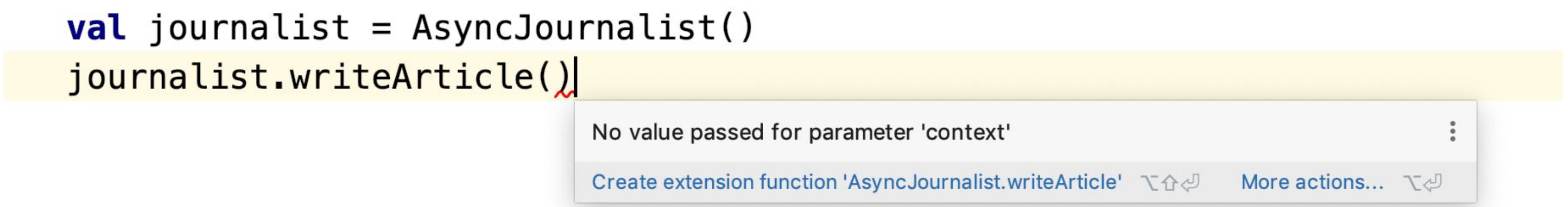
ASYNCHRONOUS CODE

```
class AsyncJournalist {

    def writeArticleAsync(): Future[String] =
                                   Future.apply("The Best Article")

}
```

ASYNCHRONOUS CODE

HOW TO USE IT IN KOTLIN?

ASYNCHRONOUS CODE



```kotlin
val journalist = AsyncJournalist()
journalist.writeArticle()
```

No value passed for parameter 'context'

Create extension function 'AsyncJournalist.writeArticle' ⌥⇧⏎
More actions... ⌥⏎

ASYNCHRONOUS CODE

```
val context = scala.concurrent.ExecutionContext.global()

val journalist = AsyncJournalist()

journalist.writeArticle(context)
```

ASYNCHRONOUS CODE

```
val context = scala.concurrent.ExecutionContext.global()

val journalist = AsyncJournalist()

journalist.writeArticle(context)
        .map({"$it :)"}, context)
        .foreach({ println(it)}, context)
```

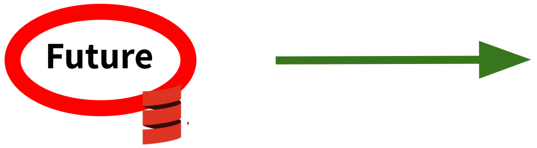
ASYNCHRONOUS CODE

HOW TO CONVERT TO DEFERRED?

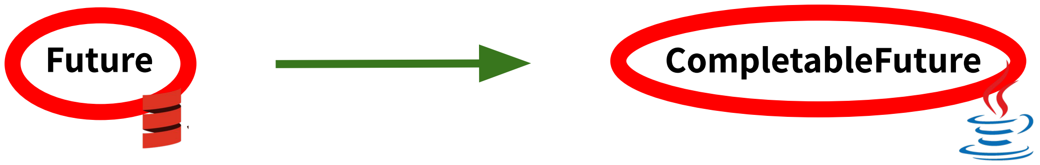
ASYNCHRONOUS CODE

Future

ASYNCHRONOUS CODE



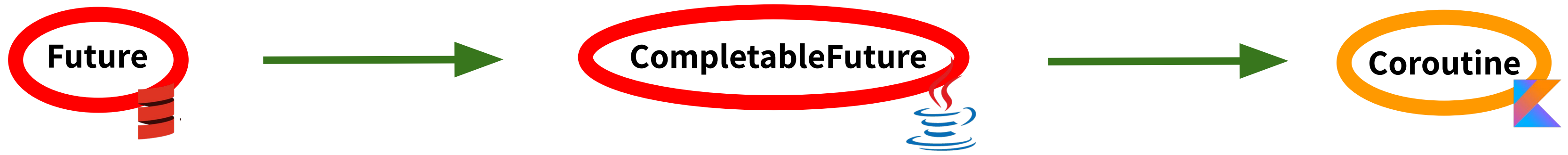
ASYNCHRONOUS CODE



ASYNCHRONOUS CODE

$$\begin{tikzcd}\text{Future} \arrow[r] & \text{CompletableFuture} \arrow[r] & {}\end{tikzcd}$$

ASYNCHRONOUS CODE



ASYNCHRONOUS CODE

```
val context = scala.concurrent.ExecutionContext.global()

val journalist = AsyncJournalist()

val article = FutureConverters.
          asJava(journalist.writeArticle(context))
          .await()

println("$article :)")
```

LET'S ADD EXTENSION

```
private fun <T>Future<T>.asKotlin() =
                FutureConverters.asJava(this)
```

ASYNCHRONOUS CODE

```kotlin
val context = scala.concurrent.ExecutionContext.global()

val journalist = AsyncJournalist()

val article = journalist.writeArticle(context)
            .asKotlin()
            .await()

println("$article :)")
```

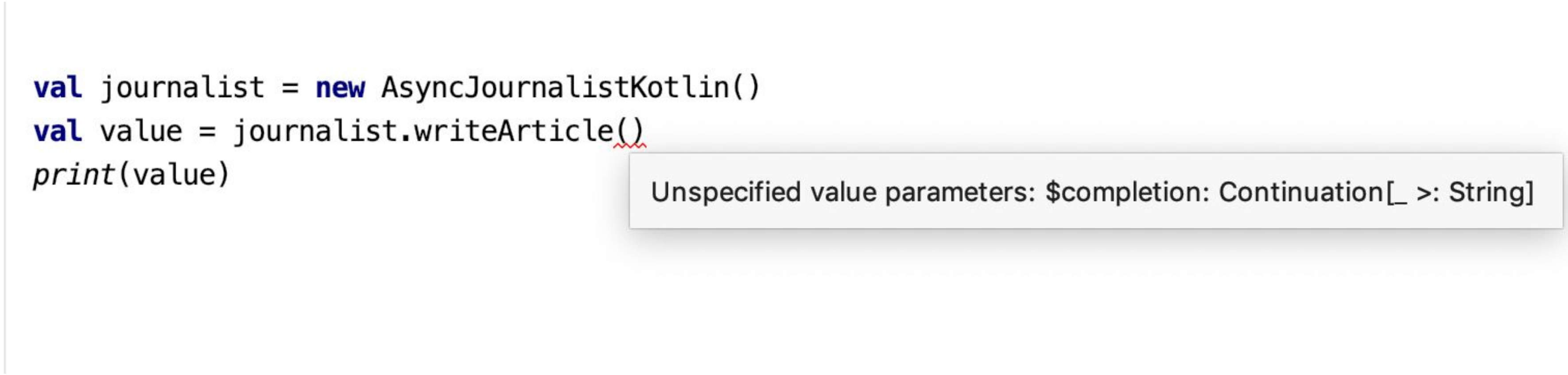
ASYNCHRONOUS CODE

COROUTINES FROM SCALA

ASYNCHRONOUS CODE

```
class AsyncJournalistKotlin {

    suspend fun writeArticle(): String {
        delay(2000)
        return "Best Article Ever... The END!"
    }
}
```

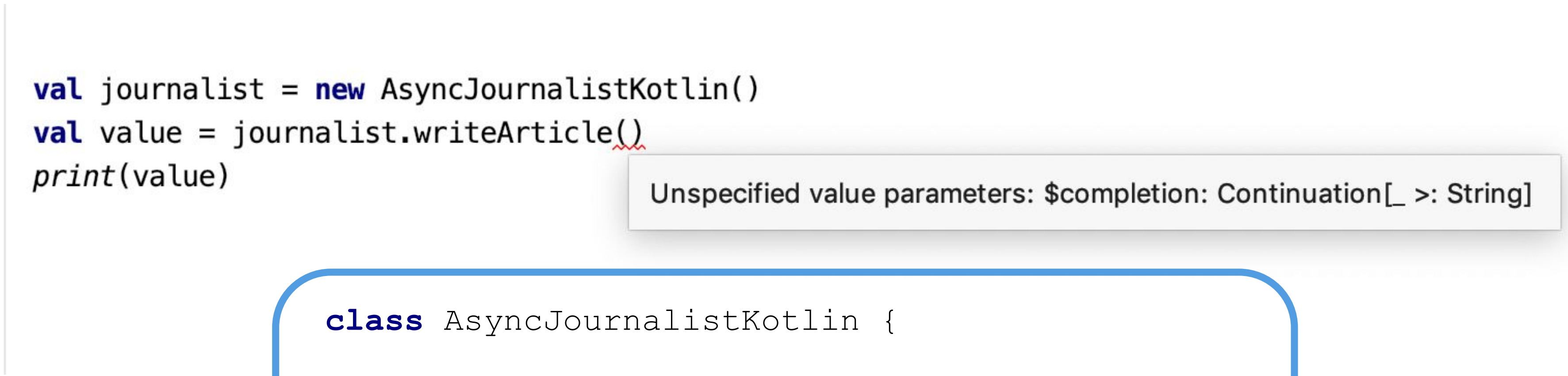
ASYNCHRONOUS CODE



```
val journalist = new AsyncJournalistKotlin()
val value = journalist.writeArticle()
print(value)
```

Unspecified value parameters: $completion: Continuation[_ >: String]

ASYNCHRONOUS CODE



```
val journalist = new AsyncJournalistKotlin()
val value = journalist.writeArticle()
print(value)
```

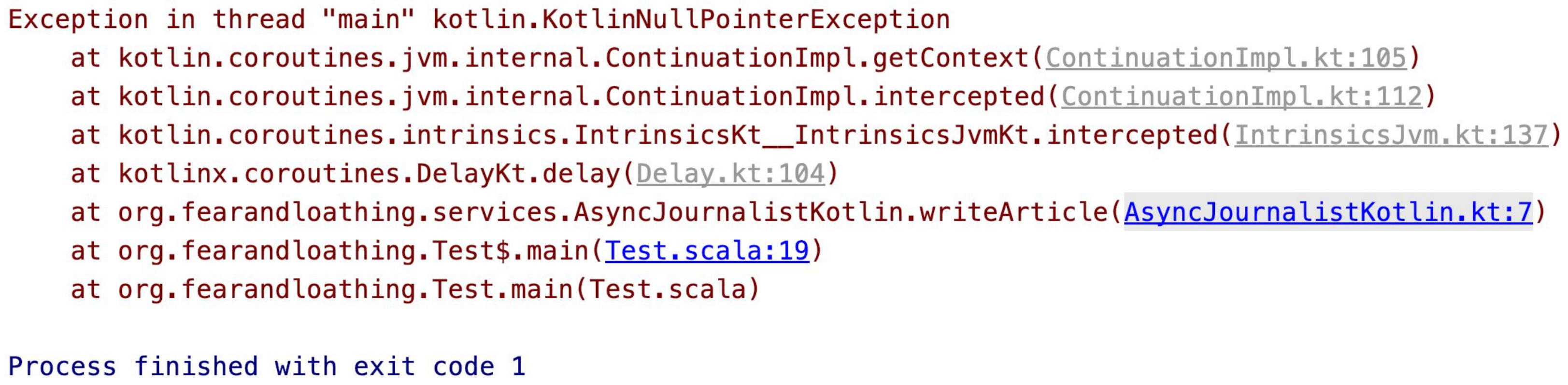
Unspecified value parameters: $completion: Continuation[_ >: String]

```
class AsyncJournalistKotlin {

    suspend fun writeArticle(): String {
        delay(2000)
        return "Best Article Ever... The END!"
    }
}
```

ASYNCHRONOUS CODE

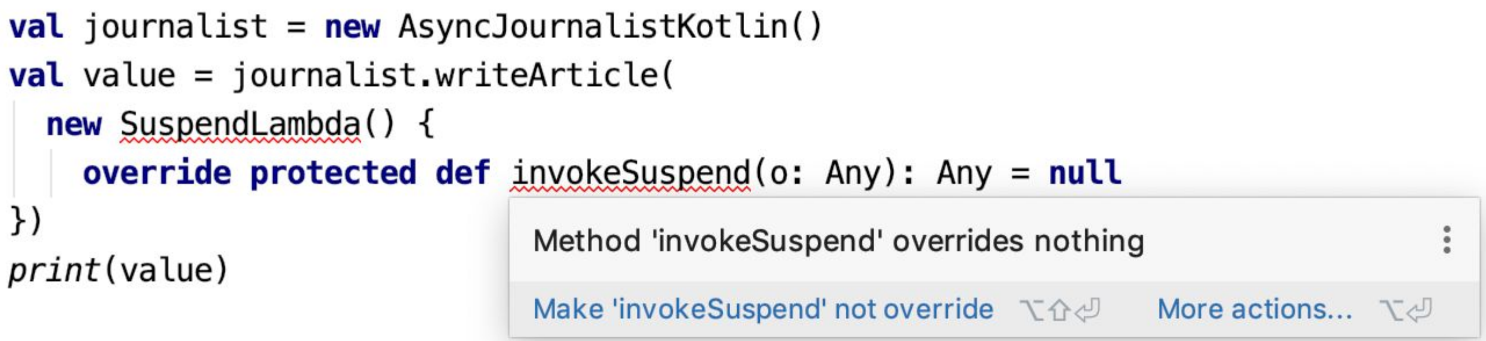
```
val journalist = new AsyncJournalistKotlin()

val value = journalist.writeArticle(null)

print(value)
```

ASYNCHRONOUS CODE



```
Exception in thread "main" kotlin.KotlinNullPointerException
    at kotlin.coroutines.jvm.internal.ContinuationImpl.getContext(ContinuationImpl.kt:105)
    at kotlin.coroutines.jvm.internal.ContinuationImpl.intercepted(ContinuationImpl.kt:112)
    at kotlin.coroutines.intrinsics.IntrinsicsKt__IntrinsicsJvmKt.intercepted(IntrinsicsJvm.kt:137)
    at kotlinx.coroutines.DelayKt.delay(Delay.kt:104)
    at org.fearandloathing.services.AsyncJournalistKotlin.writeArticle(AsyncJournalistKotlin.kt:7)
    at org.fearandloathing.Test$.main(Test.scala:19)
    at org.fearandloathing.Test.main(Test.scala)

Process finished with exit code 1
```

ASYNCHRONOUS CODE



```
val journalist = new AsyncJournalistKotlin()
val value = journalist.writeArticle(
  new SuspendLambda() {
    override protected def invokeSuspend(o: Any): Any = null
})
print(value)
```

Method 'invokeSuspend' overrides nothing

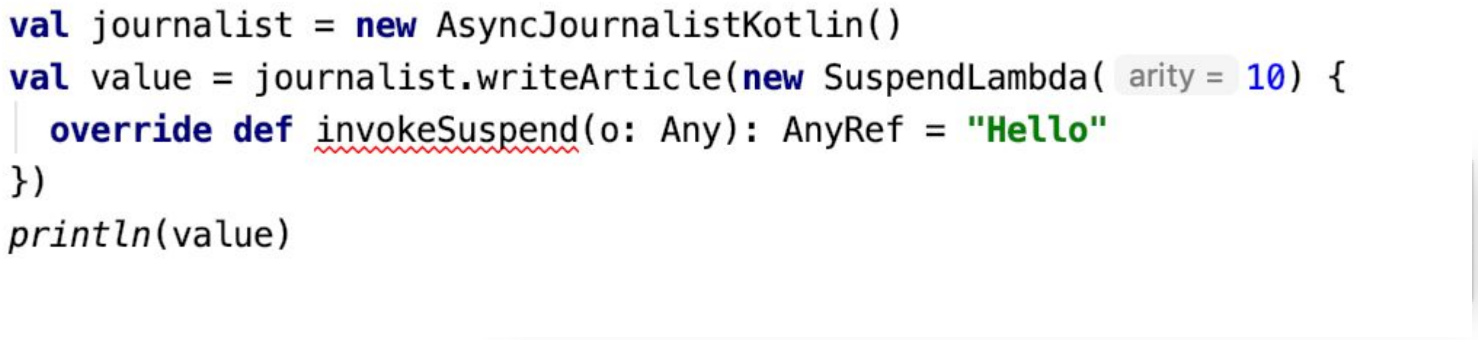
ASYNCHRONOUS CODE

```
Error:(34, 42) overloaded method constructor SuspendLambda
with alternatives:
   (x$1: Int)kotlin.coroutines.jvm.internal.SuspendLambda
<and>
   (x$1: Int,x$2:
kotlin.coroutines.Continuation[Object])kotlin.coroutines.j
vm.internal.SuspendLambda
 cannot be applied to ()
    val v1 = journalist.writeArticle(new SuspendLambda() {
```

ASYNCHRONOUS CODE

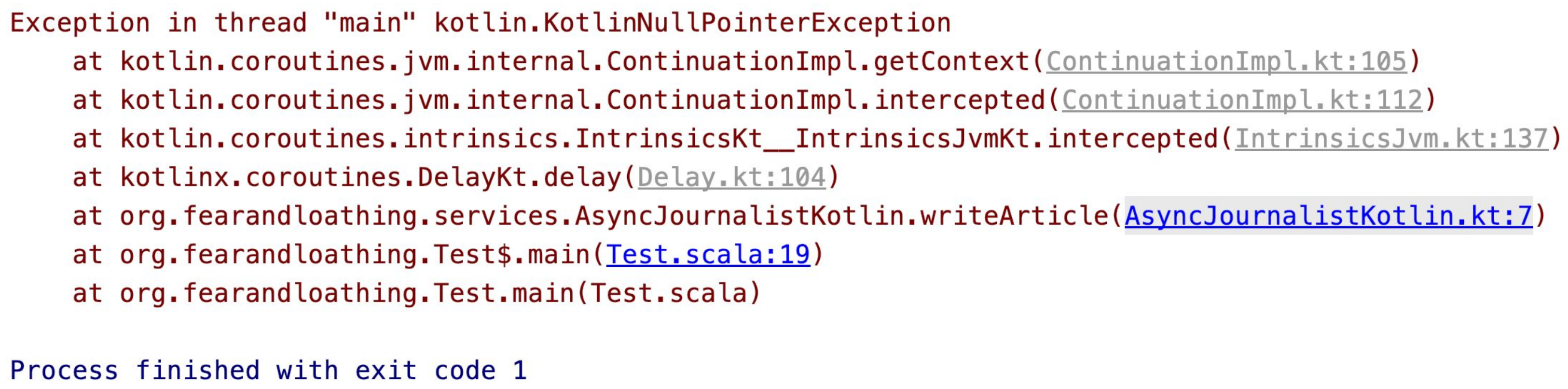
```
Error:(34, 42) overloaded method constructor SuspendLambda
with alternatives:
   (x$1: Int)kotlin.coroutines.jvm.internal.SuspendLambda
<and>
   (x$1: Int,x$2:
kotlin.coroutines.Continuation[Object])kotlin.coroutines.j
vm.internal.SuspendLambda
 cannot be applied to ()
     val v1 = journalist.writeArticle(new SuspendLambda() {
```

ASYNCHRONOUS CODE



```
val journalist = new AsyncJournalistKotlin()
val value = journalist.writeArticle(new SuspendLambda( arity = 10) {
  override def invokeSuspend(o: Any): AnyRef = "Hello"
})
println(value)
```

ASYNCHRONOUS CODE



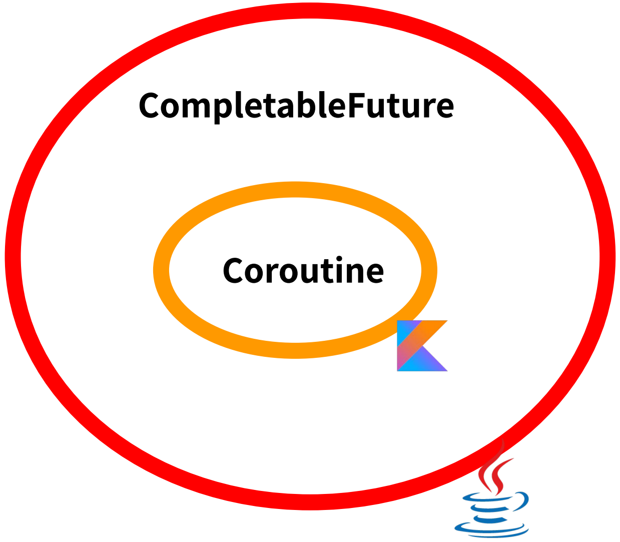
```
Exception in thread "main" kotlin.KotlinNullPointerException
    at kotlin.coroutines.jvm.internal.ContinuationImpl.getContext(ContinuationImpl.kt:105)
    at kotlin.coroutines.jvm.internal.ContinuationImpl.intercepted(ContinuationImpl.kt:112)
    at kotlin.coroutines.intrinsics.IntrinsicsKt__IntrinsicsJvmKt.intercepted(IntrinsicsJvm.kt:137)
    at kotlinx.coroutines.DelayKt.delay(Delay.kt:104)
    at org.fearandloathing.services.AsyncJournalistKotlin.writeArticle(AsyncJournalistKotlin.kt:7)
    at org.fearandloathing.Test$.main(Test.scala:19)
    at org.fearandloathing.Test.main(Test.scala)

Process finished with exit code 1
```

STOP IT!!
COMEDY CENTRAL

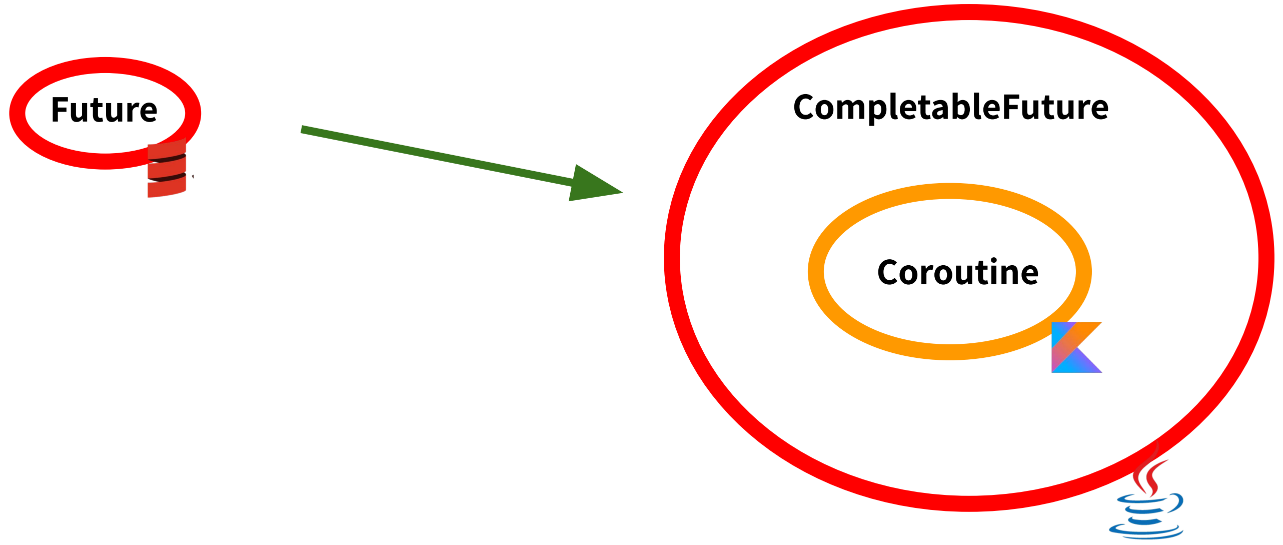
ASYNCHRONOUS CODE

Coroutine

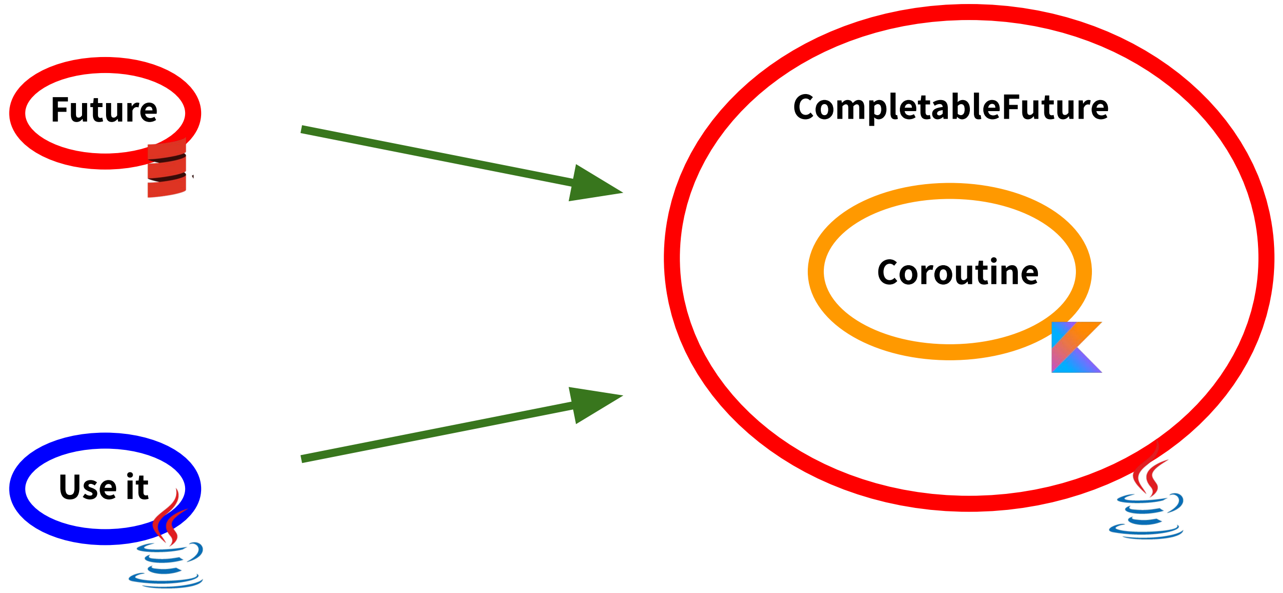
ASYNCHRONOUS CODE



ASYNCHRONOUS CODE



ASYNCHRONOUS CODE



ASYNCHRONOUS CODE

```kotlin
class AsyncJournalistKotlin {

    suspend fun writeArticle(): String {
        delay(2000)
        return "Best Article Ever... The END!"
    }
    fun writeArticleAsync(): CompletableFuture<String> =
                GlobalScope.future {
                    writeArticle()
                }

}
```

ASYNCHRONOUS CODE

```
val journalist = new AsyncJournalistKotlin()

journalist.writeArticleAsync()
    .thenAccept(s => println(s"Article: $s"))
```

ASYNCHRONOUS CODE

```
final var journalist = new AsyncJournalistKotlin();

journalist.writeArticleAsync()
    .thenAccept(s -> System.out.println("Article: " + s));
```

PROBLEM 8

CONTEXT

SWITCH

IS

PAINFUL

04

TIPS & TRICKS

INTEROP ALGORITHM

1. Split into modules
2. Define compile order
3. Try to use it as is
4. Find compiled code
5. Decompile (if needed)
6. Use it as from Java
7. Search for Java interop
8. Compile

05

SUMMARY

SUMMARY

- **No source level interop**
- **Bytecode level interop**
- **Kotlin can be integrated to existing Scala project**
- **IDE support is not that bad**
- **Don’t be afraid of mixed projects**

06

LINKS

LINKS

https://docs.scala-lang.org

https://github.com/leveretkafear-and-loathing

https://kotlinlang.org/docs/reference/java-interop.html

THANKS

Photos by JR Korpa on Unsplash

NOW I WILL TAKE
YOUR QUESTIONS.

CONTACT ME

Marharyta Nedzelska

Software Engineer @ Wix

KKUG & KotLand Kyiv

Speaker

https://medium.com/@margoqueen95

@jMargaritaN twitter

THE END

CONTEST ANSWERS

IMPLICIT EXAMPLE

```
class Test {

    private implicit val B: Int = 5

    def f(a: Int) (implicit b: Int): Int = {
        a + b
    }

    fun main(args: Array[String]): Unit = {
      new Test().f(2) // 2 + 5 = 7
    }
}
```

IMPLICIT EXAMPLE

```
import org.fearandloathing.dto.Converter;


val entity = new Comments(/* some code here */ );

val dto = Converter.convertCommentEntity()
                   .convert(entity);
```

EXTENSION FUNCTIONS

```
@file:JvmName("CommentsHelper")
package org.fearandloathing.entity

                                          ...

fun Comments.copy(): Comments {

                                          ...
}
```