Scala Basics

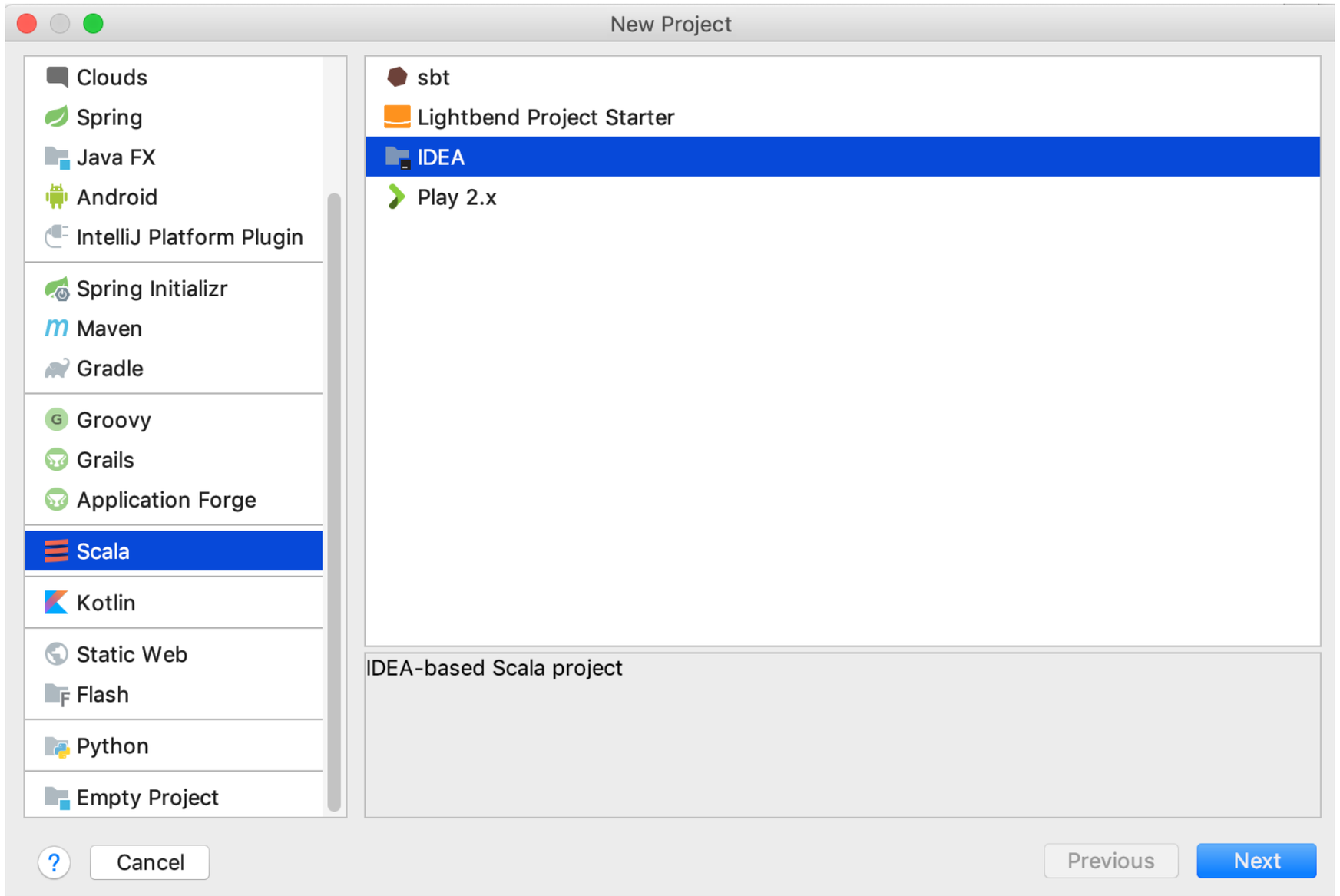
Scala Basics

```scala
package example

object Hello {

  def main(args: Array[String]): Unit = {
    println("Hello Scala!")
  }
}
```

A first example in Scala

Prints "Hello Scala!" to the screen

Scala Basics



Project setup

Create new IDEA Scala project in IntelliJ

Scala Basics

```
package example
```

```
object Hello {

  def main(args: Array[String]): Unit = {
    println("Hello Scala!")
  }
}
```

Package declaration

Must match the directory structure in the src directory

This file is saved in the directory "src/example"

To create example, right click the src directory > new > package

Scala Basics

```scala
package example

object Hello {

  def main(args: Array[String]): Unit = {
    println("Hello Scala!")
  }
}
```

Objects

Objects can store variables and functions*

Name must match it's filename

This code is in the file "src/example/Hello.scala"

*We call them methods when they are declared using the "def" keyword

Scala Basics

```
package example

object Hello {
  def main(args: Array[String]): Unit = {
    println("Hello Scala!")
  }
}
```

Main Method

The method that executes when you run your object/program

Will always have this header

Scala Basics

```
package example

object Hello {

  def main(args: Array[String]): Unit = {
    println("Hello Scala!")
  }
}
```

Print Line

Prints "Hello Scala!" to the screen

This slide intentionally left blank

```
package example

object FirstObject {

  def multiplyByTwo(input: Double): Double = {
    input * 2.0
  }

  def main(args: Array[String]): Unit = {
    var x: Double = 7.0
    var result = multiplyByTwo(x)
    println(result)
  }

}
```

Next Level

Prints 14.0 to the screen

```
package example

object FirstObject {

  def multiplyByTwo(input: Double): Double = {
    input * 2.0
  }

  def main(args: Array[String]): Unit = {
    var x: Double = 7.0
    var result = multiplyByTwo(x)
    println(result)
  }

}
```

Functions

Must declare types!

This function takes a Double as a parameter and returns a Double

Returned value is the last expression that's evaluated

```
package example

object FirstObject {

  def multiplyByTwo(input: Double): Double = {
    input * 2.0
  }

  def main(args: Array[String]): Unit = {
    var x: Double = 7.0
    var result = multiplyByTwo(x)
    println(result)
  }

}
```

Variables

Type declaration optional, but helpful

Create a mutable* variable x of type Double and initialize if to the value 7.0

*Value can change

```
package example

object FirstObject {

  def multiplyByTwo(input: Double): Double = {
    input * 2.0
  }

  def main(args: Array[String]): Unit = {
    var x: Double = 7.0
    var result = multiplyByTwo(x)
    println(result)
  }

}
```

Variables

Variable declaration without a type

Type is inferred by the return type of the function

This slide intentionally left blank

```scala
package example

object Conditional {

  def computeSize(input: Double): String = {
    val large: Double = 60.0
    val medium: Double = 30.0
    if (input >= large) {
      "large"
    } else if (input >= medium) {
      "medium"
    } else {
      "small"
    }
  }

  def main(args: Array[String]): Unit = {
    println(computeSize(70.0))
    println(computeSize(50.0))
    println(computeSize(10.0))
  }

}
```

Prints:

large
medium
small

```scala
package example

object Conditional {

  def computeSize(input: Double): String = {
    val large: Double = 60.0
    val medium: Double = 30.0
    if (input >= large) {
      "large"
    } else if (input >= medium) {
      "medium"
    } else {
      "small"
    }
  }

  def main(args: Array[String]): Unit = {
    println(computeSize(70.0))
    println(computeSize(50.0))
    println(computeSize(10.0))
  }

}
```

Conditional

No return statements

The conditional determines which expression evaluates last

```scala
package example

object Conditional {

  def computeSize(input: Double): String = {
    val large: Double = 60.0
    val medium: Double = 30.0
    if (input >= large) {
      "large"
    } else if (input >= medium) {
      "medium"
    } else {
      "small"
    }
  }

  def main(args: Array[String]): Unit = {
    println(computeSize(70.0))
    println(computeSize(50.0))
    println(computeSize(10.0))
  }

}
```

Conditional

No return statements

The conditional determines which expression evaluates last

Project Setup Demo

Lecture Question

- In a package named "lecture" create an object named "FirstObject" with a method named "computeShippingCost" that takes a Double representing the weight of a package as a parameter and returns a Double representing the shipping cost of the package

- The shipping cost is ($)5 + 0.25 for each pound over 30
    - Every package weighing 30 pounds or less will cost 5 to ship
    - A package weighing 31 pounds cost 5.25 to ship
    - A package weighing 40 pounds cost 7.50 to ship
- Submit a zip file of your project to AutoLab: File > Export to zip file

* Later lecture questions won't be available until lecture