Scala Basics

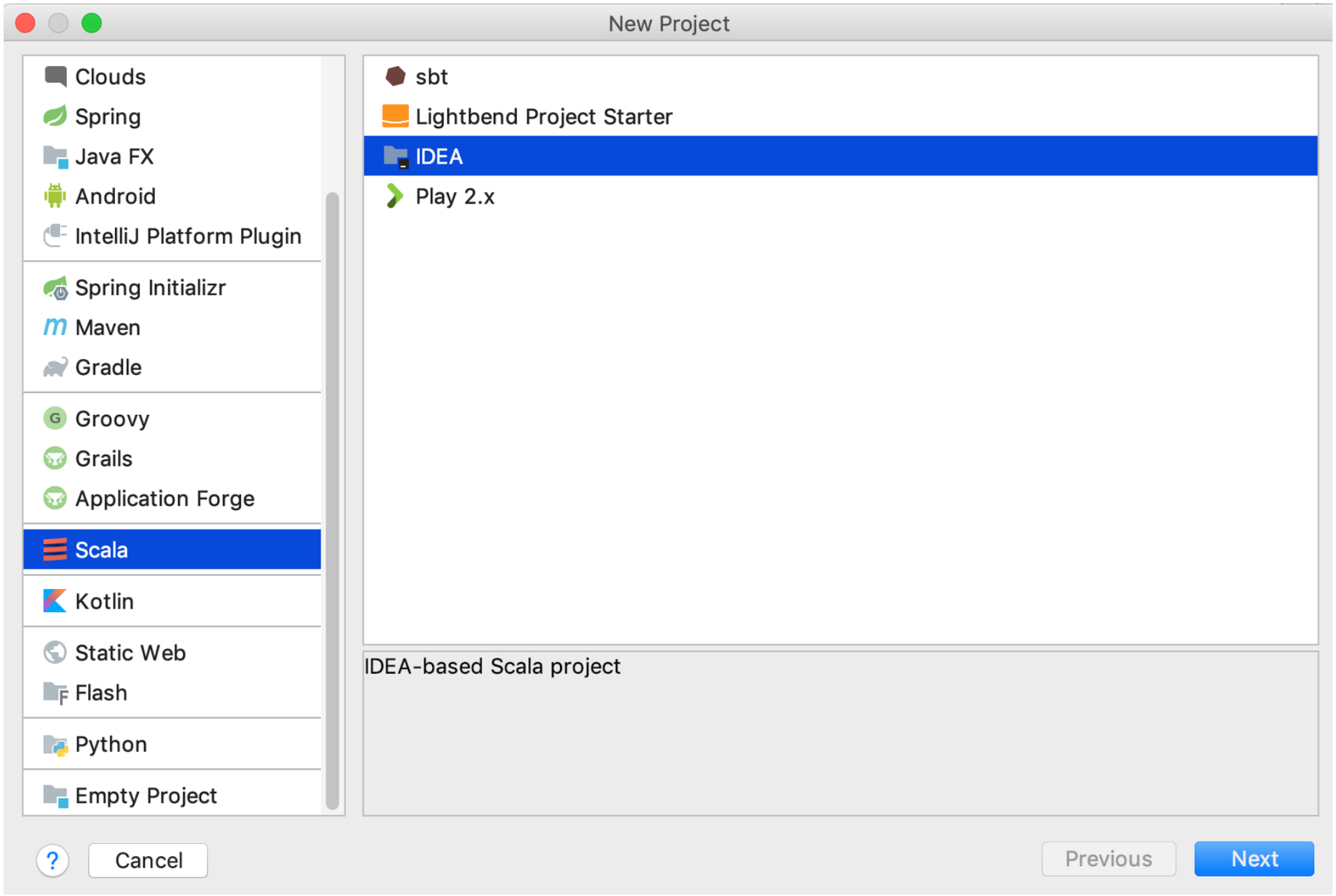
Lecture Question

- 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 per pound over 30
    - Every package weighing 30 pounds or less will cost 5 to ship
    - Ex. A package weighing 31 pounds cost 5.25 to ship
    - Ex. A package weighing 40 pounds cost 7.50 to ship
    - Ex. A package weighing 30.4 pounds cost 5.10 to ship
- Submit a zip file of your project to AutoLab: File > Export to zip file

Hello World

Scala Basics



Project setup

Create new IDEA Scala project in IntelliJ

Scala Basics

```
package week1.basics

object Hello {

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

A first example in Scala

Prints "Hello Scala!" to the screen

Scala Basics

```
package week1.basics
```

```
object Hello {

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

}
```

Package declaration

Should match the directory structure in the src directory

This file is saved in the directory "src/week1/basics"

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

Scala Basics

```
package week1.basics

object Hello {

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

Objects

Objects can store variables and functions*

Name should match the filename

This code is in the file "src/week1/basics/Hello.scala"

*We call them methods when they are part of an object

Scala Basics

```scala
package week1.basics

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 the header "**def** main(args: Array[String]): Unit"

Scala Basics

```
package week1.basics

object Hello {

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

Print Line

Prints "Hello Scala!" to the screen

Methods

Variables

```
package week1.basics

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)
  }

}
```

Prints 14.0 to the screen

```
package week1.basics

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)
  }

}
```

Methods

Must declare types!

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

```
package week1.basics

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)
  }

}
```

Methods

Must explicitly define the **type** of each parameter

Parameter name and type are separated by a colon :

```
package week1.basics

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)
  }

}
```

Methods

The return **type** must also be explicit

Return type follows the parameter list

Parameter list and return type are separated by a colon :

```
package week1.basics

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)
  }

}
```

Methods

The body of the method is executed when the method is called

Return statements are optional (And discouraged)

The returned value is the last expression that's evaluated during the method call

```
package week1.basics

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

Creates a mutable* variable named **x** of type Double and initializes it to the value 7.0

*Value can change

```
package week1.basics

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 method

Conditionals

```scala
package week1.basics

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

```
package week1.basics

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))
  }

}
```

Values declared with **val** cannot change

Reassignment causes an error

```
package week1.basics

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))
  }

}
```

Conditionals

Similar to JavaScript syntax

```
package week1.basics

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))
  }

}
```

Conditionals and Methods

The return value of this method is determined by the conditional

The conditional determines which expression evaluates last
-Any code after the conditional would break this example

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 per pound over 30
    - Every package weighing 30 pounds or less will cost 5 to ship
    - Ex. A package weighing 31 pounds cost 5.25 to ship
    - Ex. A package weighing 40 pounds cost 7.50 to ship
    - Ex. A package weighing 30.4 pounds cost 5.10 to ship
- Submit a zip file of your project to AutoLab: File > Export to zip file