Objects and Classes

Lecture Question - Makeup

In Python. Submit under Makeup_Monday - February 4

Function: In a python package named "lecture" create a file named "FirstObject" with a function named "computeShippingCost" that takes a float representing the weight of a package as a parameter and returns a float representing the shipping cost of the package

The shipping cost is ($)5 + 0.25 for each pound over 30

Unit Testing: In a package named "tests" create a class/file named "UnitTesting" as a test suite that tests the computeShippingCost method

* This question will be open until midnight

Objects

- State / Variables
  - Objects store their state in variables
  - [Vocab] Often called fields, member variables, or instance variable
- Behavior / Functions
  - Objects contains functions that can depend on its state
  - [Vocab] When a function is part of an object it's called a **method**

Object With State

```scala
package oop_classes

object ObjectWithState {

  // State of the object
  var x: Int = 10

  // Behavior of the object
  def doubleX(): Unit = {
    this.x *= 2
  }

}
```

```scala
package oop_classes

object ObjectMain {

  def main(args: Array[String]): Unit = {
    ObjectWithState.doubleX()
    println(ObjectWithState.x)
  }

}
```

- Any variable outside of all methods is part of the state of that object
- State can be altered from within the object or from other objects
- Keyword **this** stores a reference to the enclosing object

Every **value** in Scala is an **object**

Classes

- Template for creating objects
  - Objects are **instantiated** from classes
- Used to create many objects
  - Each object can have a different state
  - Each has its own copies of the instance variables

Simple Class

```scala
package oop_classes

class IntWrapper {

  var x:Int = 0

  def doubleX(): Unit = {
    this.x *= 2
  }

}
```

```scala
package oop_classes

object IntWrapperMain {

  def main(args: Array[String]): Unit = {
    val one: IntWrapper = new IntWrapper
    val two: IntWrapper = new IntWrapper

    println(one.x)
    one.x = 7
    println(one.x)
    one.doubleX()

    // one and two have different internal states
    println(one.x)
    println(two.x)
  }

}
```

- Class defines the fields and methods of its objects
- Use the **new** keyword to create objects from a class

Constructor

- Method called to create a new object
- Sets the initial state of the object
- [Scala] All parameters become member variables
  - Use var in constructor so the state can change

```scala
package oop_classes

class Point2D(var x: Double, var y: Double) {

}


 val p1: Point2D = new Point2D(3, 6)
```

Example

```
package oop_classes

class Point2D(var x: Double, var y: Double) {

}
```

```
package oop_classes

object PointMain {

  def main(args: Array[String]): Unit = {
    val p1: Point2D = new Point2D(3, 6)
    p1.x = 5

    println("(" + p1.x + ", " + p1.y + ")")
  }

}
```

- Create and modify an instance from a class
- Alter and access the objects state

Classes

- Int, Double, Boolean, List, Array, Map
  - Are all classes
  - We use these classes to create values

```
var list: List[Int] = List(2, 3, 4)
```

- Create objects by calling the classes constructor
- List is setup in a way that we don't use **new**
- For our classes we will use the **new** keyword

A Note on Access Modifiers

- Determine who (which classes/objects) can alter state and control behavior of an object
- Access modifiers are Controversial
- Communities around different languages cannot agree on these

Access Modifiers

- If you're familiar with **Java** you're familiar with these
  - public / private / protected
  - default is package private
- In **Scala**
  - private / protected
  - default is public
- In **Python**
  - No access modifiers
  - Everything is public
- In **JavaScript**
  - No access modifiers
  - Everything is public
  - Can create work-arounds to simulate private variables

Accessor/Mutator

- Common in some languages to make all member variables private
  - Java
  - C++
- State is never used directly from outside the object
- Use accessor and mutator methods instead
  - Or getter and setter

```java
package oop_classes;

public class AccessModifiers{

    // NOTE: This is Java code

    private int x;

    public int getX(){
        return this.x;
    }

    public void setX(int x){
        this.x = x;
    }
}
```

Lecture Question

Question: In a package named "oop" create a Scala class named "Score" with the following:

- A constructor that takes an Int and stores it in a member variable named score
- A method named scoreGoal that takes no parameters and has return type Unit that increments the score by 1
- A method named isWinner that takes a Score object as a parameter and returns a Boolean. The method returns true if this instances score is strictly greater than the inputs objects score

* This question will be open until midnight