Q1 Template

```
abstract class Player(var level: Int) {

 def strength(): Int

 def levelUp(): Unit = {
   this.level += 1.0
 }

}

class Warrior(warriorLevel: Int) extends Player(warriorLevel) {

 override def strength(): Int= {
   this.level * 3
 }

}

class Wizard(wizardLevel: Int) extends Player(wizardLevel) {

 override def strength(): Int= {
   this.level * 1
 }

 override def levelUp(): Unit = {
   this.level += 3
 }

}

// Part 1
def main(args: Array[String]): Unit = {
 val warrior: Warrior= new Warrior(1)
 val wizard: Wizard= new Wizard(2)
 val player: Player= wizard

 println(warrior.strength())
 warrior.levelUp()
 println(warrior.strength())

 println(wizard.strength())
 wizard.levelUp()
 println(wizard.strength())
 player.levelUp
println(player.strength())
}
```

```
// Part 2
object Part2 {
def totalStrength(team: List[Player]): Int= {
  var total = 0
  for(player <- team){
    total += player.strength()
   }
  total
}

def levelUpTeam(team: List[Player]): Unit = {
   for(player <- team){
   player.levelUp()
  }
}

def main(args: Array[String]): Unit = {
  val warrior: Warrior= new Warrior(2)
  val wizard: Wizard= new Wizard(5)
  val player: Player = wizard

  val team: List[Player] = List(warrior, wizard, player, new Warrior(4))

  println(totalStrength(team))
  levelUpTeam(team)
  println(totalStrength(team))
  }

}
```

WRITE OUT ALL THE PRINT STATEMENTS FOR PART 1 and 2

Q2

```
class Model {

 var score: Int = 0

 def displayField(): Double = {
   this.score
 }

 def reward(value: Int): Unit = {
   this.score += value
 }


///////////scroll
```

```
 def hit(value: Int): Unit = {
   if(this.score <= 5) {
     this.score += value/2
   }
   else {
     this.score += value
   }
 }

 def destroy(): Unit = {
   if (this.score > 0) {
     this.score -= 10
   }
   else {
     this.score = 0
   }
 }

}

class Controller(model:Model) {

 def reward(event: ActionEvent): Unit = model.reward(2)

 def hit(event: ActionEvent): Unit = model.hit(-2)

 def destroy(event: ActionEvent): Unit = model.destroy()

 def userAction(event: KeyEvent): Unit = {
   event.getCode.getName match {
     case "U" => model.reward(4)
     case "D" => model.hit(-4)
     case "J" => model.reward(6)
     case "K" => model.hit(-5)
     case "X" => model.destroy()
     case _ => model.hit(-3)
   }
 }
}

class QuizButton(display: String, action: EventHandler[ActionEvent]) extends Button {

 val size = 200
 minWidth = size
 minHeight = size
 onAction = action
 text = display
 style = "-fx-font: 30 ariel;"

}
```

```
object View extends JFXApp {

 val model: Model = new Model()
 val controller: Controller = new Controller(model)

 var textField: TextField = new TextField {
   editable = false
   style = "-fx-font: 26 ariel;"
   text.value = model.displayField().toString
 }

 stage = new PrimaryStage {
   title = "Quiz GUI"
   scene = new Scene() {
     content = List(
       new GridPane {
         add(textField, 0, 0, 2, 1)
         add(new QuizButton("reward", controller.reward), 0, 1)
         add(new QuizButton("hit", controller.hit), 1, 1)
         add(new QuizButton("destroy", controller.destroy), 2, 1)
       }
     )
   }

   addEventFilter(KeyEvent.KEY_PRESSED, controller.userAction)

   // update the display after every event
   addEventFilter(Event.ANY, (event: Event) => textField.text.value =
model.displayField().toString())
 }

}
```

Q2

For each part below, state what is displayed in the app's text field after the sequence of inputs. You should assume that the app is restarted before each new sequence is executed.

a) The user clicks the following buttons: “reward”, “reward”, “reward”, “hit”, “hit”
b) The user presses “U”, “U”, “D”, “D”, “X”, “P”
c) The user clicks the following buttons: “reward”, “hit”, “hit”, “destroy”, “reward”, “destroy”
d) The user does the following: presses “K”, clicks “hit”, clicks “reward”, clicks “reward”, presses “J”
e) The user does the following: clicks “hit”, clicks “destroy”, presses “J”, presses “L”, presses “X”