# Lab Quiz 2 Template (Section B4)

## TA Instructions

Before Lab:

- Copy this template into a new file and put it in the Lab Quiz 2 folder with your lab section in the name. Do not edit this template directly since other TAs will need to copy this
- Fill in all the personalizations for each question template **before your lab section**
    - Feel free to alter the questions beyond the templates as long as you're testing the same concepts. The template provides a minimal amount of variance between labs
- Have a plan to display the quiz to your students
    - Setup a github repo and have students view the code on their computers with close supervision to ensure they only access the quiz code
    - -or- Notify Jesse via Slack that you want your quiz printed

During Lab:

- Give them 1 hour to complete the Quiz. You don't have to start at the beginning of lab but you must end the quiz 1 hour after you start
- **Team Adjustments:** For the remaining time, address any team reformations that may be necessary due to resignations and non-participants. Use your discretion in rearranging teams and bring up any tough calls in the Slack channel for help deciding.

Grading:

- Start grading as the quiz ends. When you're done grading, add their scores into AutoLab and release the grades for your section. Their grades should be posted within 24 hours of taking the quiz. You can hand the quizzes back next week
- Run your code to be 100% sure of the answers

## Student Instructions

- This is an exam environment
    - No talking
    - No material allowed except blank paper and writing utensils
    - Only ask TA's questions clarifying what a question is asking

# Q1

```
abstract class Employee(var salary: Double) {

 def pay(): Double

 def jobReview(): Unit = {
   this.salary += 200.0
 }

}

class Teacher(salary: Double) extends Employee(salary) {

 override def pay(): Double = {
   this.salary += 100.0
 }

}

class Janitor(salary: Double) extends Employee(salary) {

 override def pay(): Double = {
   this.salary + 25.0
 }

 override def jobReview(): Unit = {
   this.salary += 15.0
 }

}

def accumulate(Employees: List[Employee]): Double = {
 var total = 0.0
 for(employee <- Employees){
   total += employee.pay()
 }
 total
}

def performJobReviews(Employees: List[Employee]): Unit = {
 for(employee <- Employees){
   employee.jobReview()
 }
}
```

Part 1) What is printed to the screen?

```
	def main(args: Array[String]): Unit = {
		val teacher = new Teacher(20000)
		val janitor = new Janitor(5000)

		println(teacher.pay())
		println(janitor.pay())
		teacher.jobReview()
		println(teacher.salary)
		janitor.jobReview()
		println(janitor.salary)
	}
```

Part 2)

```
	def main(args: Array[String]): Unit = {
		val employeeList: List[Employee] = List(new Janitor(5000), new Teacher(20000),
                                                new Janitor(8000), new Teacher(35000))

		println(accumulate(employeeList))
		performJobReviews(employeeList)

		for(employee <- employeeList) {
			println(employee.salary)
		}
	}
```

# Q2

```
class Model {

 var money: Int = 10

 def displayMoney(): Double = {
   this.money
 }

 def deposit(value: Int): Unit = {
   this.money += value
}

 def empty(): Unit = {
   this.money = 0
```

```
 }

}


class Controller(model: Model) {

 def b1Pressed(event: ActionEvent): Unit = model.deposit(10)

 def b2Pressed(event: ActionEvent): Unit = model.empty()

 def userAction(event: KeyEvent): Unit = {
   event.getCode.getName match {
     case "A" => model.deposit(10)
     case "B" => model.deposit(20)
     case "C" => model.empty()
     case "D" => model.deposit(30)
     case "E" => model.deposit(40)
     case "X" => model.empty()
     case _ => model.deposit(-5)
   }
 }

}

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("Deposit $10", controller.b1Pressed), 0, 1)
          add(new QuizButton("Empty", controller.b2Pressed), 1, 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 INPUTS

- Press b1 (”Deposit $10” button)
- Press b2 (“Empty” button)
- Press ‘A’ key
- Press ‘C’ key
- Press ‘Z’ key