Stone Barrett

Assignment 05

7/21/19

Problem 1: Employees

For this program, two classes and two exception classes need to work well together. The first class, *Employee*, initializes variables for employee names, salaries, and social security numbers. It also creates getters and setters for those values, as well as methods to determine whether the SSN entered is appropriate (throwing exceptions *SSNCharacterException* and *SSNLengthException*) and converting information to a string to print. The second class, *EmployeeBuilder*, sets up a scanner and interfaces with a user to gain information about the employee(s). It checks whether the employee record entered already exists. It also calculates the average salary and returns whether an employee is above or below that value.

Screenshots:

ConferenceFeeCalculator.java | *Employee.java | *SSNLengthException.java

```java
// Stone Barrett
// Assignment 5 - Problem 1

// Open class
public class Employee
{
    // Initialization
    private String name;
    private double salary;
    private String ssnNumber;

    // Constructor
    public Employee(String name, double salary, String ssnNumber) throws SSNCharacterException, SSNLengthException
    {
        setName(name);
        setSalary(salary);
        setSSNNumber(ssnNumber);
    }

    // Mutators
    public void setName(String name)
    {
        this.name = name;
    }

    public void setSalary(double salary)
    {
        this.salary = salary;
    }

    public void setSSNNumber(String ssnNumber) throws SSNCharacterException, SSNLengthException
    {
        validateSSNNumber(ssnNumber);
        this.ssnNumber = ssnNumber;
    }

    // Validate SSN
    private void validateSSNNumber(String ssnNumber) throws SSNCharacterException, SSNLengthException
    {
        int count = 0;

        for(int i = 0; i < ssnNumber.length(); i++)
        {
            if(ssnNumber.charAt(i) >= '0' && ssnNumber.charAt(i) <= '9')
                count++;
            else if((ssnNumber.charAt(i) != '-') && (ssnNumber.charAt(i) != ' '))
                throw new SSNCharacterException(ssnNumber);
        }

        if(count != 9)
        {
            throw new SSNLengthException(ssnNumber);
        }
    }
    // Accessors
    public String getName()
    {
        return name;
    }

    public double getSalary()
    {
        return salary;
    }

    public String getSSNNumber()
    {
        return ssnNumber;
    }

    public boolean equals(Employee other)
    {
        return((name.equalsIgnoreCase(other.name)) || (ssnNumber.equalsIgnoreCase(other.ssnNumber)));
    }

    // Preparing information to be displayed
    public String toString()
    {
        return("SSN: " + ssnNumber + " Name: " + name + " Salary: " + String.format("%.2f", salary));
    }
}
```

ConferenceFeeCalculator.java | *Employee.java | *SSNLengthException.java | *SSNCharacterException.java | *EmployeeBuilder.java

```java
// Stone Barrett
// Assignment 5 - Problem 1

public class SSNLengthException extends Exception
{
    public SSNLengthException(String ssn)
        {
            super("SSN entered: " + ssn + ".\nSSN should contain 9 digits (no dashes, spaces, or characters)");
        }
}
```

ConferenceFeeCalculator.java | *Employee.java | *SSNLengthException.java | *SSNCharacterException.java | *EmployeeBuilder.java

```java
// Stone Barrett
// Assignment 5 - Problem 1

public class SSNCharacterException extends Exception
{
    public SSNCharacterException(String ssn)
    {
        super("SSN entered: " + ssn + ".\nSSN should contain 9 digits (no dashes, spaces, or characters)");
    }
}
```

ConferenceFeeCalculator.java | *Employee.java | *SSNLengthException.java | *SS

```java
// Stone Barrett
// Assignment 5 - Problem 1

// Importing scanner
import java.util.Scanner;

// Open class
public class EmployeeBuilder
{
    // Check if employee number already in use
    private static boolean checkEmployeeEquality(Employee employees[], Employee emp, int count)
    {
        for(int i = 0; i < count; i++)
        {
            if(employees[i].equals(emp))
                return true;
        }
        return false;
    }

    // Calculate average salary of all employees
    private static double averageSalary(Employee employees[], int count)
    {
        double total = 0;

        for(int i = 0; i < count; i++)
        {
            total += employees[i].getSalary();
        }

        if(count > 0)
            return(total/count);
        else
            return 0;
    }

    // Main function
    public static void main(String[] args)
    {
        Employee employees[] = new Employee[100];
        int count = 0;
        String choice, name, ssnNumber;

        Scanner scan = new Scanner(System.in);

        double salary;

        // Loop until user enters "n" or array reaches 100 entries
        do
        {
            // UI, prompting for information
            System.out.println("Enter details for Employee " + (count + 1) + ": ");
            System.out.print("Name: ");
            name = scan.nextLine();
            System.out.print("SSN: ");
            ssnNumber = scan.nextLine();
            System.out.print("Salary: ");
            salary = scan.nextDouble();
            scan.nextLine();
```

```java
            try
            {
                employees[count] = new Employee(name, salary, ssnNumber);

                // Check if employee already exists
                if(!checkEmployeeEquality(employees, employees[count], count))
                    count++;
                else
                    System.out.println("Employee with same name or SSN already exists");
            } catch(SSNCharacterException | SSNLengthException e)
                {
                    System.out.println(e.getMessage());
                }

                // Prompt for more employees
                System.out.print("Do you want to enter more records? (y/n)");
                choice = scan.nextLine();

        }while((choice.equalsIgnoreCase("y")) && (count < 100));

        double avg = averageSalary(employees, count);

        // Display all records
        System.out.println("Details of the Employees : ");

        for(int i = 0; i < count; i++)
        {
            if(employees[i].getSalary() < avg)
                System.out.println(employees[i] + " Salary below average");
            else
                System.out.println(employees[i] + " Salary above average");
        }
        scan.close();
    }
}
```

Problem 2 (Bonus): Registration

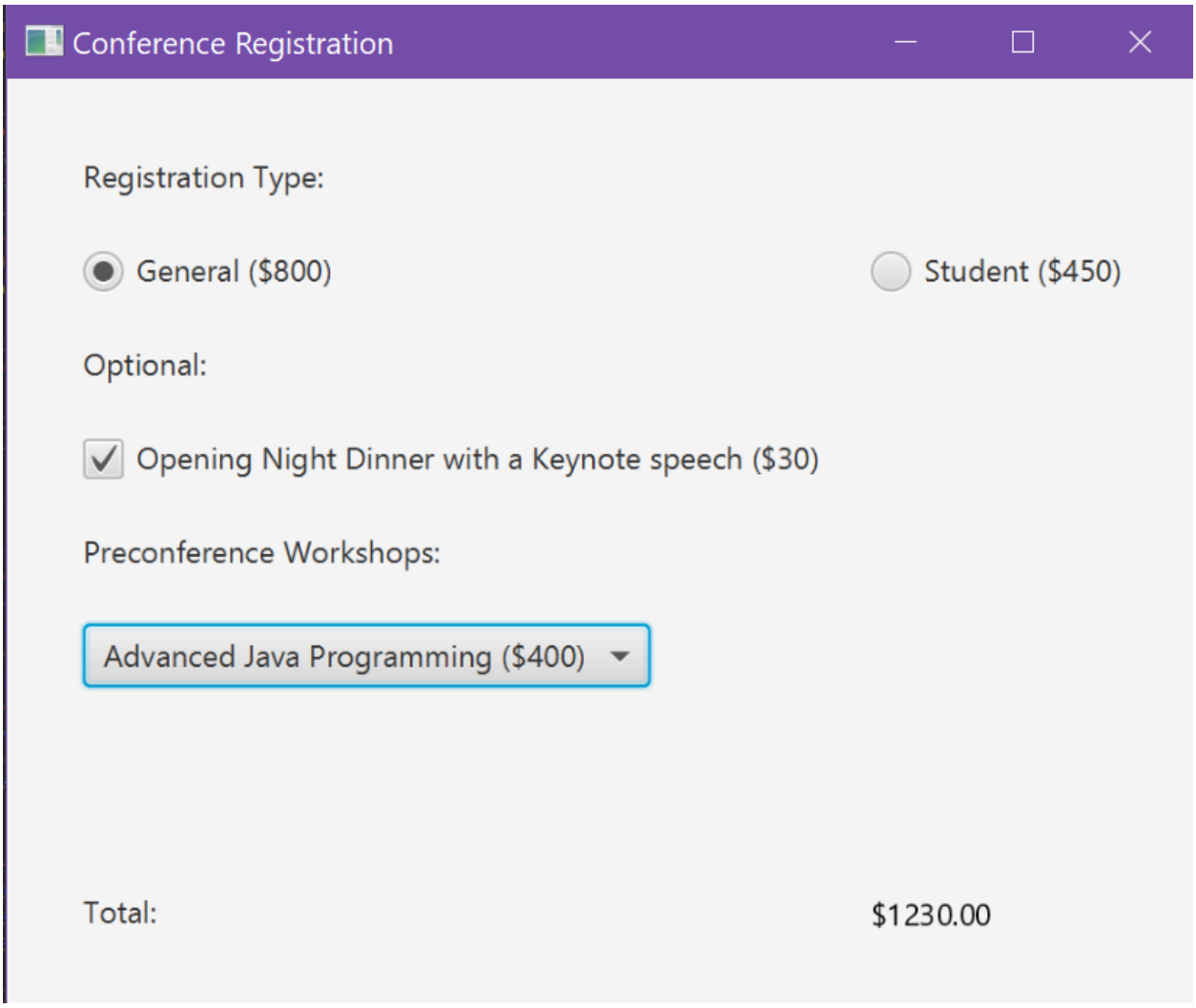
For this bonus problem, only one class is required but it is longer than the previous problem's classes. This program imports all JavaFX libraries needed and is granted access to the JavaFX folder in the program files. It initializes the UI elements, including the buttons, check box, and drop-down menu. The stage and gridpane are set up and everything is positioned. The radio buttons are put into a toggle group so that only one can be selected at one time. The checkbox can be selected and deselected. The drop-down menu offers the different choices and one can be selected. The cost is updated whenever selections are made, then displayed in the bottom, right corner of the gridpane.

Screenshots:

ConferenceF... | Tip.java | CollectionO... | ElectricCus... | *InputPane.java

```java
// Stone Barrett
// Assignment 5 - Problem 2 (Bonus)

// Importing libraries
import javafx.application.Application;
import static javafx.application.Application.launch;
import javafx.collections.FXCollections;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.CheckBox;
import javafx.scene.control.ChoiceBox;
import javafx.scene.control.Label;
import javafx.scene.control.RadioButton;
import javafx.scene.control.ToggleGroup;
import javafx.scene.layout.GridPane;
import javafx.scene.text.Text;
import javafx.stage.Stage;

// Open class inherit from Application library
public class ConferenceFeeCalculator extends Application
{
    // Initializing UI elements
    RadioButton genReg, stuReg;
    ToggleGroup group;
    CheckBox dinner;
    Text totalText;
    ChoiceBox<String> precon;

    // String array holding topics
    String[] precon_names = {"Introduction to E-commerce ($350)", "The Future of the Web ($300)",
        "Advanced Java Programming ($400)", "Network Security ($450)"};

    // Double array holding conference costs
    double[] precon_costs = {350, 300, 400, 450};

   @Override

    public void start(Stage primaryStage)
    {
        // Creating grid pane
        GridPane root = new GridPane();

        // Spacing
        root.setHgap(20);
        root.setVgap(20);
        root.setPadding(new Insets(30));

        // User data
        genReg = new RadioButton("General ($800)");
        genReg.setUserData(800.0);

        // Updating
        genReg.setOnAction(e -> update());
        stuReg = new RadioButton("Student ($450)");
        stuReg.setUserData(450.0);
        stuReg.setOnAction(e -> update());
```

```java
        // Toggle group, only one can be selected
        group = new ToggleGroup();
        genReg.setToggleGroup(group);
        stuReg.setToggleGroup(group);
        genReg.setSelected(true);
        dinner = new CheckBox("Opening Night Dinner with a Keynote speech ($30)");
        dinner.setUserData(30.0);
        dinner.setOnAction(e -> update());

        // Creating choice box
        precon = new ChoiceBox<>(FXCollections.observableArrayList(precon_names));

        // Updating
        precon.getSelectionModel().selectedIndexProperty().addListener(e->update());

        totalText = new Text();

        // Adding components to root
        root.add(new Label("Registration Type: "), 0, 0);
        root.add(genReg, 0, 1);
        root.add(stuReg, 1, 1);
        root.add(new Label("Optional: "), 0, 2);
        root.add(dinner, 0, 3);
        root.add(new Label("Preconference Workshops: "), 0, 4);
        root.add(precon, 0, 5);
        root.add(new Label("Total: "), 0, 9);
        root.add(totalText, 1, 9);

        // Setting scene and displaying
        Scene scene = new Scene(root);
        primaryStage.setScene(scene);
        primaryStage.setTitle("Conference Registration");
        primaryStage.show();

        // Updating
        update();
    }

    // Event handler
    private void update()
    {
        // Total cost
        double total = 0;

        // Adding radio button selection to total
        total += (Double) group.getSelectedToggle().getUserData();

        // Adding other selections to total
        if (dinner.isSelected())
        {
            total += (Double) dinner.getUserData();
        }

        // Getting choice box selection
        int index=precon.getSelectionModel().getSelectedIndex();

        // Adding choice box selection to total
        if(index!=-1)
        {
            total+=precon_costs[index];
        }

        // Final cost
        totalText.setText(String.format("$%.2f", total));
    }

    public static void main(String[] args)
    {
        launch(args);
    }
}
```


Conference Registration
Registration Type:
General ($800)
Student ($450)
Optional:
Opening Night Dinner with a Keynote speech ($30)
Preconference Workshops:
Advanced Java Programming ($400)
Total:
$1230.00