

## Lab 4

# *Classes and Objects*

Name: \_\_\_\_\_

---

### Prerequisites

Please complete the following prerequisite activities before beginning this lab:

- Read Chapter 4 of *Java Software Solutions*
- Review the process of creating objects using the new operator
- Review the process of invoking service methods of objects
- Review the purpose of the visibility modifiers
- Review the purpose of the static modifier

---

### Introduction

Chapter 4 introduced several fundamental object-oriented concepts. The more practice you get using these concepts in the development of programs, the more natural they will become. This lab walks you through the development of an application that uses objects to play roulette.

---

### Lab Objectives

After completing this lab, you should understand and be able to describe the following:

- Object creation
- Instance data
- Method invocation
- Static data and methods
- Visibility modifiers

---

### Activities

Perform the following activities, in order, recording any information requested in the space provided. Your instructor will provide any specific information necessary to complete this lab.

## Developing Objects

- 1 Enter the following program, saving it in a file called `Roulette.java`, or copy it from the location specified by your instructor:

```
import java.io.*;
import java.util.*;

//*****
//  Class Roulette contains the main driver for a roulette game.
//*****
class Roulette {

    //=====
    //  Contains the main processing loop for the roulette game.
    //=====
    public static void main (String[] args) throws IOException {
        Player player = new Player (100);
        boolean done = false;

        System.out.println ("Cash available: " + player.cash());
        System.out.println();
        while (!done) {
            System.out.println ("Cash available: " + player.cash());
            done = !player.spin_again();
            System.out.println();
        }
    } // method main

} // class Roulette

//*****
//  Class Player represents one roulette player.
//*****
class Player {
    private int bet, money;
    private BufferedReader stdin;

    //=====
    //  The Player constructor sets up the initial available cash.
    //=====
    public Player (int initial_money) {
        money = initial_money;
        stdin = new BufferedReader (new InputStreamReader(System.in));
    } // constructor Player
}
```

```

//=====
// Returns this player's current available cash.
//=====
public int cash() {
    return money;
} // method cash

//=====
// Prompts the user and reads betting information.
//=====
public void make_bet() throws IOException {
    System.out.print ("How much to bet: ");
    bet = Integer.parseInt (stdin.readLine());
    money = money - bet;
} // method make_bet

//=====
// Determines if the player wants to spin the wheel again.
//=====
public boolean spin_again() throws IOException {
    String answer;
    System.out.print ("Spin again [y/n]? ");
    answer = stdin.readLine();
    return (answer.equals("y") || answer.equals("Y"));
} // method spin_again

} // class Player

//*****
// Class Wheel represents a roulette wheel and its operations. Its
// data and methods are static because there is only one wheel.
//*****
class Wheel {

    final static int RED = 1;
    final static int BLACK = 2;
    final static int NUMBER = 3;
    private static int ball_position;
    private static int color;
    private final static int POSITIONS = 37;

//=====
// Presents betting options and reads player choice.

```

```

//=====
public static int bet_options (BufferedReader stdin)
    throws IOException {
    System.out.println ("1. Bet on red");
    System.out.println ("2. Bet on black");
    System.out.println ("3. Bet on a particular number");
    System.out.print ("Your choice? ");
    return Integer.parseInt (stdin.readLine());
}

} // class Wheel

```

- 2 In its present form, the program simply loops as long as the player responds that they want to spin again. Carefully trace the processing as it currently exists. Compile and execute the program.
- 3 At the beginning of the loop inside the main method, insert a call to the `make_bet` method of the player object. At this point the `make_bet` method simply reads the amount of the bet and deducts it from the available funds. Compile and test the program. Why does the main method use the player method called `cash` to determine how much cash is available?
- 4 We need a check to ensure that the amount bet is less than or equal to the amount available. One way to handle a bet that is too high is to print an error message and reprompt, but in this program we'll simply assume in that case that the amount bet is equal to the available cash. Insert that check in the `make_bet` method. Print the message "Betting it all!" if appropriate. Compile and test the program, stressing both possibilities.
- 5 Note that there is currently no call to the method `bet_options` of the `Wheel` class, which prints the betting possibilities and reads the user's choice. Create a new instance variable called `bet_type` in the `Player` class. Then insert a call to the `bet_options` method from the `make_bet` method, storing the result in `bet_type`. Note that the method `bet_options` is static, therefore its invocation is through the `Wheel` class, not an instance. In fact, we'll never instantiate the `Wheel` class. It is designed assuming there is only one roulette wheel in the game, and therefore all data in the class is static. Compile and test this modification.
- 6 We need to make one more modification to the `make_bet` method. If the user wants to bet on a particular number, we must prompt for and read that number. Store it in a variable called `number` in the player object. Use the constant `Wheel.number` to test for that `bet_type`.
- 7 Note that some issues, like the bet options, are presented by the `Wheel` class, while others, like determining the number to bet, are handled by the `Player` class. Why is this?

- 8 We need to be able to spin the roulette wheel. A typical roulette wheel has 37 positions, numbered from 0 to 36. Create a static Random object in the Wheel class. Then create a method called spin in the Wheel class that determines and sets the ball\_position variable. Insert a call to the spin method in the main loop after the player makes their bet.
- 9 The even numbers on the roulette wheel are colored red, and the odd numbers are colored black. In the spin method, use the remainder operator % to determine the color of the current ball position, and set the color variable of the Wheel class. Make it correspond to the RED and BLACK constants already defined in the Wheel class. At the end of the spin method, print the results of the spin, both color and number. Compile and test your program.
- 10 Create a method called payoff in the Wheel class, which accepts parameters representing the bet amount, the bet type, and the number bet. Note that the third parameter is not needed if the bet is on color. The purpose of this method is to return the amount won by the player. Because there are two types of payoff categories, we will define two support methods (private to the Wheel class) to help us determine the payoff amount. If the bet is on a particular number, invoke a method called number\_payoff, passing in the bet amount and the number. Otherwise, invoke a method called red\_black\_payoff, passing in the bet amount and the bet type.
- 11 Create the method number\_payoff, such that if the number bet matches the number on the wheel, the payoff is 36 times the bet. Otherwise the payoff is zero. Return the payoff amount.
- 12 Create the method red\_black\_payoff, such that if the color bet matches the color on the wheel, the payoff is 2 times the bet. Otherwise the payoff is zero. Return the payoff amount.
- 13 Create a method called payment in the Player class that determines the winnings using the payoff method, prints the payoff amount, and adds it to the available cash. Insert a call to payment in the main processing loop after the wheel is spun. Compile and test the program.
- 14 Test the system thoroughly. Note that the chances of betting on a number and winning are relatively small. How could we modify the program to facilitate the testing of this issue while still demonstrating confidence in the program's logic?
- 15 This program could have been designed many ways. Describe an alternative design, ensuring that the roles and responsibilities of each class and object are appropriate.
- 16 Make sure your program documentation is up to date and complete. Print the final version of your program and turn it in with your lab.