CPSC 121
Project 01: Parking Finder

Project Overview

In this project, you will develop an application that randomly generates four parking spots and determines which parking spot is the closest to your current location.

Objectives

1. Write a class with a main method (ParkingFinder).
2. Use an existing, custom class (ParkingSpot).
3. Use classes from the Java standard library.
4. Work with numeric data.
5. Work with standard input and output.

Specification

Existing class that you will use: ParkingSpot.java
Class that you will create: ParkingFinder.java

Getting Started

1. Create a new Eclipse project for this assignment and import (copy/paste) ParkingSpot.java into your project.
2. Create a new Java class called ParkingFinder and add a main method.
3. Read the ParkingSpot documentation below and look through the ParkingSpot.java file to familiarize yourself with what it contains and how it works before writing any code of your own.
4. Start implementing your ParkingFinder class according to the specifications. TEST OFTEN!
5. You may need some methods from the Math class, such as Math.abs, Math.ceil, and Math.min.

ParkingSpot (provided class)

*You do **not** need to modify this class. You will just use it in your ParkingFinder driver class.*

You are given a class named ParkingSpot that keeps track of the details about a parking spot. Each parking spot instance will keep track of the following data:

- Street name of the spot.
    - The street name must be specified when the spot is created. Street names should be given based on the order they are read in (e.g. "1st Street", "2nd Street", "3rd Street", "4th Street").
- Coordinates in a two-dimensional city grid.
    - The (x, y) coordinates must be specified when the spot is created.
- Whether or not the spot is available.
    - When a new parking spot is created, it is marked as available.
- The charge per 10 minute interval.
    - The default charge is $0.25 per 10 minute interval. As with all parking meters, payment must be in whole intervals.

The ParkingSpot class has the following methods available for you to use:

1. public ParkingSpot(String street, int locationX, int location Y) – Constructor
2. public Boolean isAvailable( )
3. public Boolean setAvailable(boolean available)
4. public void setCostPerInterval(double cost)
5. public double getCostPerInterval( )
6. public String getStreet( )
7. public int getLocationX( )
8. public int getLocationY( )
9. public String toString( )

To find out what each method does, look at the documentation comments within the ParkingSpot class itself.

ParkingFinder (the driver class)

You will write a class called ParkingFinder. It is the driver class, meaning, it will contain the main method.

In this class, you will create a random starting coordinate (your car's position) and four ParkingSpot objects with random location coordinates. After your cars position is determined and the spots are created, you will calculate how much it would cost to park in each spot for a given number of time and use conditional statements to determine which spot is closest to you.

Your ParkingFinder code should not duplicate any data or functionality that belongs to the ParkingSpot objects. Make sure you are accessing data using the ParkingSpot methods. DO NOT use a loop or arrays to generate your ParkingSpots. We will discuss how to do this later, but please do not over complicate things now.

1. User Input
    a) Console input should be handled with a Scanner object. You should **never** make more than one Scanner from System.in in any program. There is only one keyboard, there should only be one Scanner.
    b) Your program will prompt the user for the following values and read the user's input from the keyboard.
        i. A long value to seed your Random number generator. The seed value read from the keyboard should be used to initialize your Random number generator. Look at the different constructors available in the Random class to recall how seed values are used.
        ii. An int value for the amount of parking time needed in minutes. You can use the nextLong( ) method of the Scanner class to read the seed for the random number generator and nextInt( ) method to read the parking time.

2. Generating the car's random location
    a. Initialize two random integers (carX and carY) in the range 0-99 inclusive to designate the driver's starting position. Use your seeded Random object to

generate the random coordinates. These must be generated before your random parking spots in order for your output to match the sample output.

   b. Print the position of the vehicle. Your output should be, x = 97, y = 37.

3. Generating random parking spots
   a. Now, create four parking spot objects at random x, y coordinates.
      i. Use the following process to create each new spot.
         - Use your existing seeded Random object to generate random x, y coordinates. The x and y coordinates must be in the range 0-99 inclusive.
         - Use the ParkingSpot constructor to instantiate a new spot with a street name and the random x, y coordinates.
      ii. Let the first two parking spots have the default cost per interval and set the cost per interval of the last two spots to $0.30 per 10 minute interval. You can set the cost per interval for a particular spot using the setCostPerInterval( ) method of the ParkingSpot class.
      iii. Before moving to the next step, print your generated parking spots to make sure everything looks correct. To print a spot, you may use the provided toString method. Look at the ParkingSpot documentation for an example of how to print a parking spot.

         Note – You should not have to store any of the data for your parking spot in variables in the main method. Each ParkingSpot instance you create will store its own data, so you can use the methods of the ParkingSpot class to get that data when you need it. For example, once you set the x location of your spot, you may retrieve it using spot1.getLocationX( );

4. Calculate the distance and charge. You must do the following for each parking spot.
   a. Calculate the distance from the driver.
      i. The distance is calculated using Manhattan geometry.
      ii. In Manhattan geometry (also known as taxicab geometry), the distance between two points (x1, y1) and (x2, y2) is the sum of the absolute value of the differences between the x and y coordinates.
   b. Calculate the cost to park in the spot for the time requested.
      i. The driver is charged per 10 minute period, so any time between 1 and 10 minutes is counted as a full 10 minutes. In other words, if the charge is $0.20 per 10 minutes, it would cost $0.40 to park for 11 minutes. You may need to check out the Math.ceil() method.
   c. Print the details of the spot along with the distance and cost to park. Format the output as shown in example below. *Note – use the toString() method of the ParkingSpot class to print the parking spot details.*

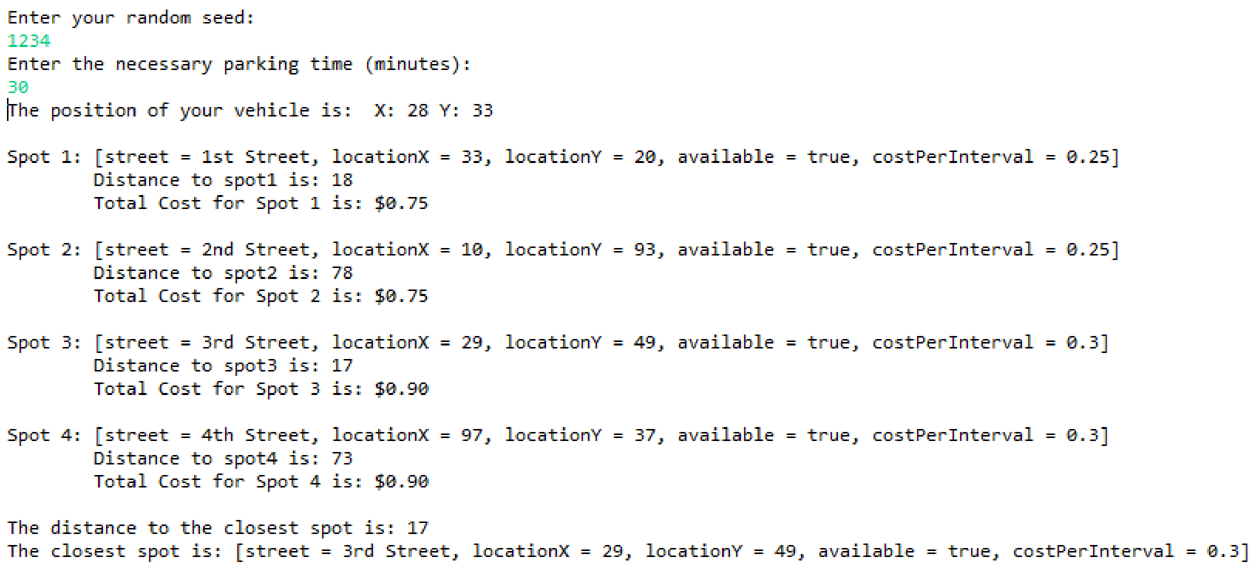
```
Spot 1: [Street = 1st Street, location = 33, location = 20, available = true, costPerInterval =
$0.25]
        Distance:  18
        Total Cost:  $0.75
```

   i. Use the NumberFormat currency formatter to print the cost.

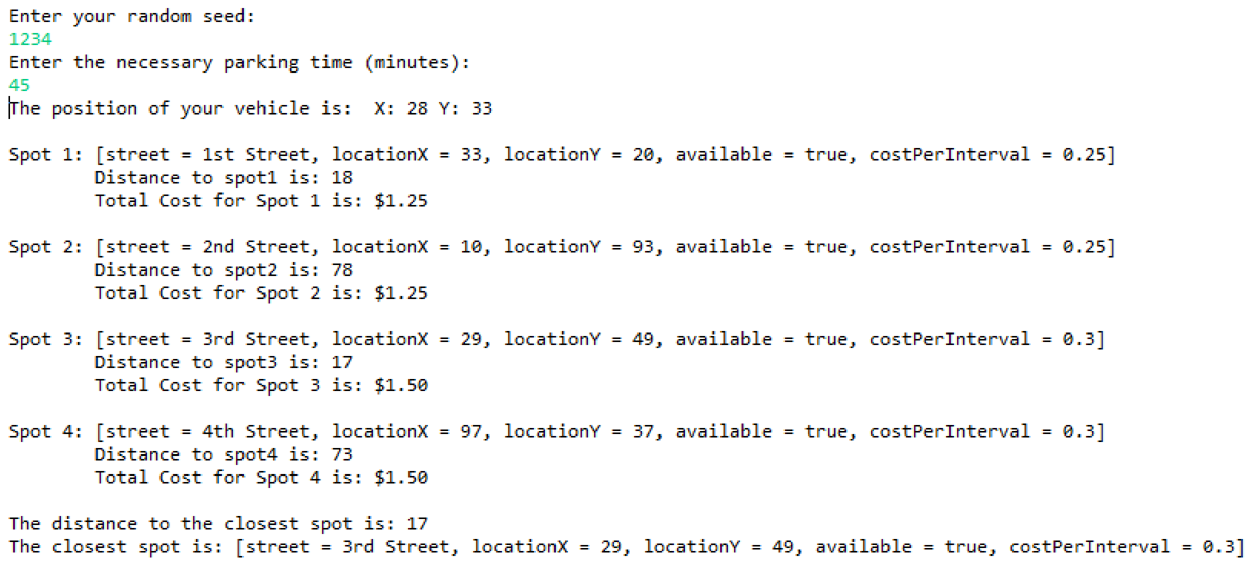
5. Find the closest spot
   a. Determine which parking spot is the closest out of the four spots.
   b. Print the distance to the closest spot and the details of the closest spot.

```
Distance to closest spot:  73
Closest spot:  [Street = 1st Street, location = 33, location = 20, available = true,
costPerInterval = $0.25]
```

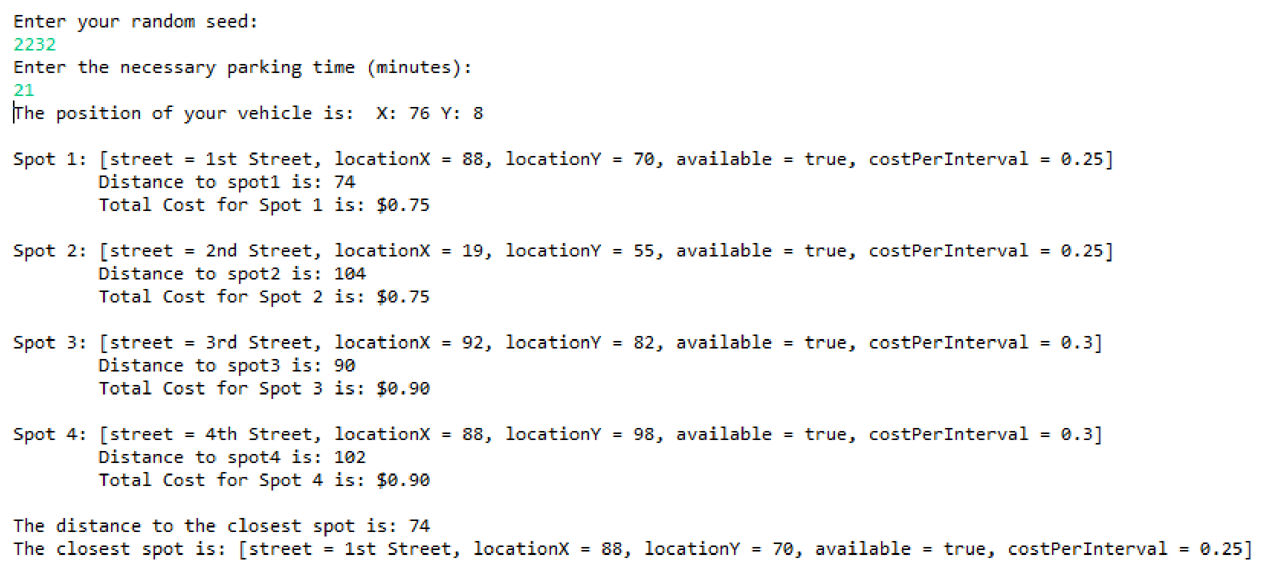
Sample Output:



```
Enter your random seed:
1234
Enter the necessary parking time (minutes):
30
The position of your vehicle is:  X: 28 Y: 33

Spot 1: [street = 1st Street, locationX = 33, locationY = 20, available = true, costPerInterval = 0.25]
        Distance to spot1 is: 18
        Total Cost for Spot 1 is: $0.75

Spot 2: [street = 2nd Street, locationX = 10, locationY = 93, available = true, costPerInterval = 0.25]
        Distance to spot2 is: 78
        Total Cost for Spot 2 is: $0.75

Spot 3: [street = 3rd Street, locationX = 29, locationY = 49, available = true, costPerInterval = 0.3]
        Distance to spot3 is: 17
        Total Cost for Spot 3 is: $0.90

Spot 4: [street = 4th Street, locationX = 97, locationY = 37, available = true, costPerInterval = 0.3]
        Distance to spot4 is: 73
        Total Cost for Spot 4 is: $0.90

The distance to the closest spot is: 17
The closest spot is: [street = 3rd Street, locationX = 29, locationY = 49, available = true, costPerInterval = 0.3]
```



```
Enter your random seed:
1234
Enter the necessary parking time (minutes):
45
The position of your vehicle is:  X: 28 Y: 33

Spot 1: [street = 1st Street, locationX = 33, locationY = 20, available = true, costPerInterval = 0.25]
        Distance to spot1 is: 18
        Total Cost for Spot 1 is: $1.25

Spot 2: [street = 2nd Street, locationX = 10, locationY = 93, available = true, costPerInterval = 0.25]
        Distance to spot2 is: 78
        Total Cost for Spot 2 is: $1.25

Spot 3: [street = 3rd Street, locationX = 29, locationY = 49, available = true, costPerInterval = 0.3]
        Distance to spot3 is: 17
        Total Cost for Spot 3 is: $1.50

Spot 4: [street = 4th Street, locationX = 97, locationY = 37, available = true, costPerInterval = 0.3]
        Distance to spot4 is: 73
        Total Cost for Spot 4 is: $1.50

The distance to the closest spot is: 17
The closest spot is: [street = 3rd Street, locationX = 29, locationY = 49, available = true, costPerInterval = 0.3]
```



```
Enter your random seed:
2232
Enter the necessary parking time (minutes):
21
The position of your vehicle is:  X: 76 Y: 8

Spot 1: [street = 1st Street, locationX = 88, locationY = 70, available = true, costPerInterval = 0.25]
        Distance to spot1 is: 74
        Total Cost for Spot 1 is: $0.75

Spot 2: [street = 2nd Street, locationX = 19, locationY = 55, available = true, costPerInterval = 0.25]
        Distance to spot2 is: 104
        Total Cost for Spot 2 is: $0.75

Spot 3: [street = 3rd Street, locationX = 92, locationY = 82, available = true, costPerInterval = 0.3]
        Distance to spot3 is: 90
        Total Cost for Spot 3 is: $0.90

Spot 4: [street = 4th Street, locationX = 88, locationY = 98, available = true, costPerInterval = 0.3]
        Distance to spot4 is: 102
        Total Cost for Spot 4 is: $0.90

The distance to the closest spot is: 74
The closest spot is: [street = 1st Street, locationX = 88, locationY = 70, available = true, costPerInterval = 0.25]
```

Submission

For this assignment, you will need to submit your ParkingFinder.java file. Additionally, I expect a README file complete with all of the items listed in the template including a section on how the program should run. For example, tell the user they need to enter a SEED, and tell them what the SEED does. Compress both the ParkingFinder.java and README file into a zip file named:
Project01_*LastName_FirstName*.zip