CS1020: Data Structures and Algorithms I

Tutorial 3 – Problem Solving with Java Library and Java Generics

(For week 5, starting 10 February 2014)

1. **[Java Random Numbers]** *Please refer to Lecture 2 slides 34-37 for this question.*

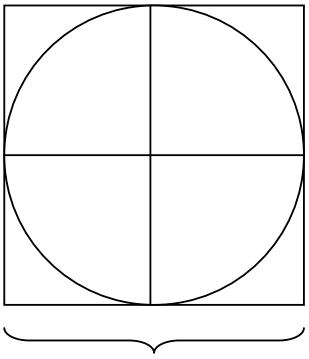
   The Monte Carlo method uses repeated random sampling to iteratively arrive at a more and more accurate answer, and is used in many scientific experiments when it is difficult to arrive at an answer through pure calculation. Read Wikipedia for a quick guide to the "Monte Carlo" method.

   The Monte Carlo simulation can be used to find the approximate value of pi (π). First, a circle of radius r is drawn within a square. As the formula for the area of a circle is πr^2, the ratio of the area of the circle to the area of the square is given by:

$$\frac{\text{Area of Circle}}{\text{Area of Square}} = \frac{\pi r^2}{(2r)^2} = \frac{\pi r^2}{4r^2} = \frac{\pi}{4}$$

   From above, it can be concluded that the ratio of the area of the circle to the area of the square is $\frac{\pi}{4}$. Therefore, π can be found by multiplying the ratio of the area of the circle to the area of the square by 4.

   The Monte Carlo simulation works by choosing multiple (x,y) coordinates from [-1.0, 1.0], and counting how many coordinates fall within the circle (this includes points that are exactly on the circumference of the circle). The circle is centered at (0, 0) and is bounded by a 2 units × 2 units square (see diagram on the right).



2 unit in length

   a. The random numbers generated by the `Random` class are considered "pseudo-random" numbers. What does this mean?

   b. Using only the `nextInt(int)` method in the `Random` class, complete the method to generate a pseudo-random real number x, such that $-1.0 \leq x \leq 1.0$.

```
public double customRandom(Random randGen) {

}
```

   c. Which `Math` method can you use to calculate the Euclidean distance between a point P at (x, y) and the origin (0, 0)?.

   d. Using the methods created or used in parts b and c, complete the code to find the approximate value of π. Run the simulation with 10 trials, 1000 trials, and 100000 trials. What happens to the accuracy of the estimation?

CS1020: Data Structures and Algorithms I

2. **[ArrayList]** Refer to the following code snippet for parts a to c:

```
import java.util.ArrayList;

public class LargestNumber {
  public static void main(String [] args) {

    ArrayList<Integer> numbers = new ArrayList<Integer>();
    numbers.add(3000);
    numbers.add(3000);
    numbers.add(1);
    numbers.add(544);
    numbers.add(64);
    numbers.add(9999);
    numbers.add(3141);
    numbers.add(733);
    numbers.add(65);

    System.out.println("Largest num: " + getLargestNumber(numbers));
    System.out.println("Index num: " +
          getIndexOfLargestNumber(numbers));
  }

  public static int getLargestNumber(ArrayList<Integer> numbers) {

    // Complete this (a)

  }

  public static int getIndexOfLargestNumber(ArrayList<Integer> numbers) {

    // Complete this (b)


  }
}
```

a. The code snippet shows an `ArrayList` of integers, added sequentially. Note that the numbers are not in any order. Using **only** the methods in the `ArrayList` class, complete the `getLargestNumber()` method to find the largest value in `numbers`. Assume that the ArrayList parameter `numbers` always contains at least 1 element.

b. Use your answer from part a to complete the `getIndexOfLargestNumber()` method to return the lowest index of the largest number.

c. The `ArrayList` API provides convenient methods to manipulate objects within an `ArrayList`. Examine the following code:

```
public class Transaction {
   private String description;
   private double amount;

   public Transaction(String description, double amount) {
      this.description = description;
      this.amount = amount;
   }

   public String getDescription() {
      return description;
   }

   public double getAmount() {
      return amount;
   }
}
```

```
import java.util.ArrayList;
public class TransactionTracker {

   private static ArrayList<Transaction> transactions;

   public static void main(String[] args) {
      transactions = new ArrayList<Transaction>();
      Transaction eggs = new Transaction("Eggs", 4.00);
      Transaction chocolate = new Transaction("Chocolate bar", 3.44);
      transactions.add(eggs);
      transactions.add(chocolate);

      Transaction eggs2 = new Transaction("Eggs", 4.00);
      int index = transactions.indexOf(eggs2);
      System.out.println(index);
   }
}
```

What will be printed out? Why is that so?

(Hint: Read the Java API on `indexOf()` method under `ArrayList` for a clue.) The solution requires a concept introduced in Lecture 3.

CS1020: Data Structures and Algorithms I

3. **[Autoboxing]** What is the output for each of the following lines? Can you explain what happened? Consider each sub-question as a separate program.

```
// Part (i)
class MyWrapperTest1 {
   private static Integer x;
   public static void main(String [] args) {
      System.out.println(x);
      System.out.println(x.toString());
   }
}
```

```
// Part (ii)
class MyWrapperTest2 {

   public static void main(String [] args) {
      Integer x = 3;
      Integer y = 2;

      int k = x % y;
      System.out.println(k);
   }
}
```

```
// Part (iii)
class MyWrapperTest3 {

   public static void main(String [] args) {
      Double finalSolution = null;
      Integer firstNum = 3000;
      Integer secondNum = 4000;

      finalSolution = firstNum / secondNum; //Integer division
      System.out.println(finalSolution);
   }
}
```

4. **[Java Generics]** Given the class `PairGen` with generics:

```
class PairGen <T> {

  private T first, second;

  public PairGen(T a, T b) {
    first = a;
    second = b;
  }

  public T getFirst() { return first; }
  public T getSecond() { return second; }

}
```

Given the class `PairNot` without generics:

```
class PairNot {

  private Object first, second;

  public PairNot(Object a, Object b) {
    first = a;
    second = b;
  }

  public Object getFirst() { return first; }
  public Object getSecond() { return second; }

}
```

   a. How would you initialize a pair of `String`, "soccer" and "football" into the `PairGen` and `PairNot` class respectively? What are the advantages of using generics?

Given the class `NewPair`:

```
class NewPair <S,T> {

  private S first;
  private T second;

  public NewPair(S a, T b) {
    first = a;
    second = b;
  }

  public S getFirst() { return first; }
  public T getSecond() { return second; }

}
```

b. Can you use the class `PairGen` to pair two objects of different data types? Why or why not?
c. Can you use the class `NewPair` to pair two objects of the same data type? Why or why not?
d. Can you use the class `NewPair` to pair two arrays of `String` and `Integer`? Why or why not?
e. Can you pass `ArrayList<String>` to a method that accepts `ArrayList<Object>`? Why or why not?

- End of Tutorial 3 -