CM2307
Object-Oriented Modelling and Programming
C1722325 – Spyros Lontos

Question 1:

i) In order to "fix" the problem I have changed some lines in the LinearCongruentialGenerator.java

```
public class LinearCongruentialGenerator implements RandomInterface {
// Generates pseudo-random numbers using:
```

I changed the code from IncompatibleRandomInterface to RandomInterface

```
public static void main(String args[]) {
// Just a little bit of test code, to illustrate use of this class.
  RandomInterface r=new LinearCongruentialGenerator();
  for (int i=0; i<10; i++) System.out.println(r.next());
```

From IncompatibleRandomInterface to RandomInterface and from r.getNextNumber() to r.next()

```
public double next() {
    seed = (a * seed + c) % m;
    return (double) seed/m;
  }
}
```

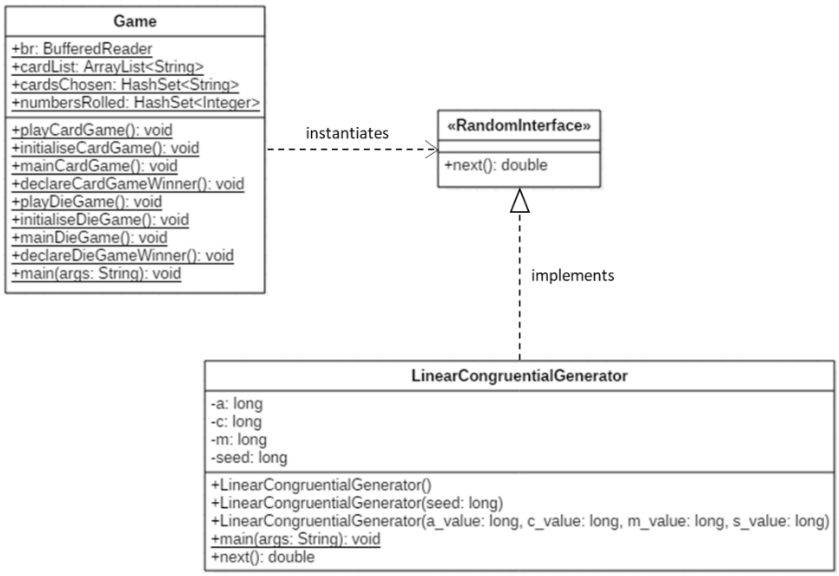
Changed the name of the operation from getNextNumber() to next().

Also fixed the Game.java file

```
// The random number generator used throughout
public static RandomInterface r =new LinearCongruentialGenerator();
```

The LinearCongruentialGenerator() was commented out and wasn't being implemented correctly to the RandomInterface. So, I simply erased the ";//"

ii) UML Diagram for Question1 after modifying the code for part (i).

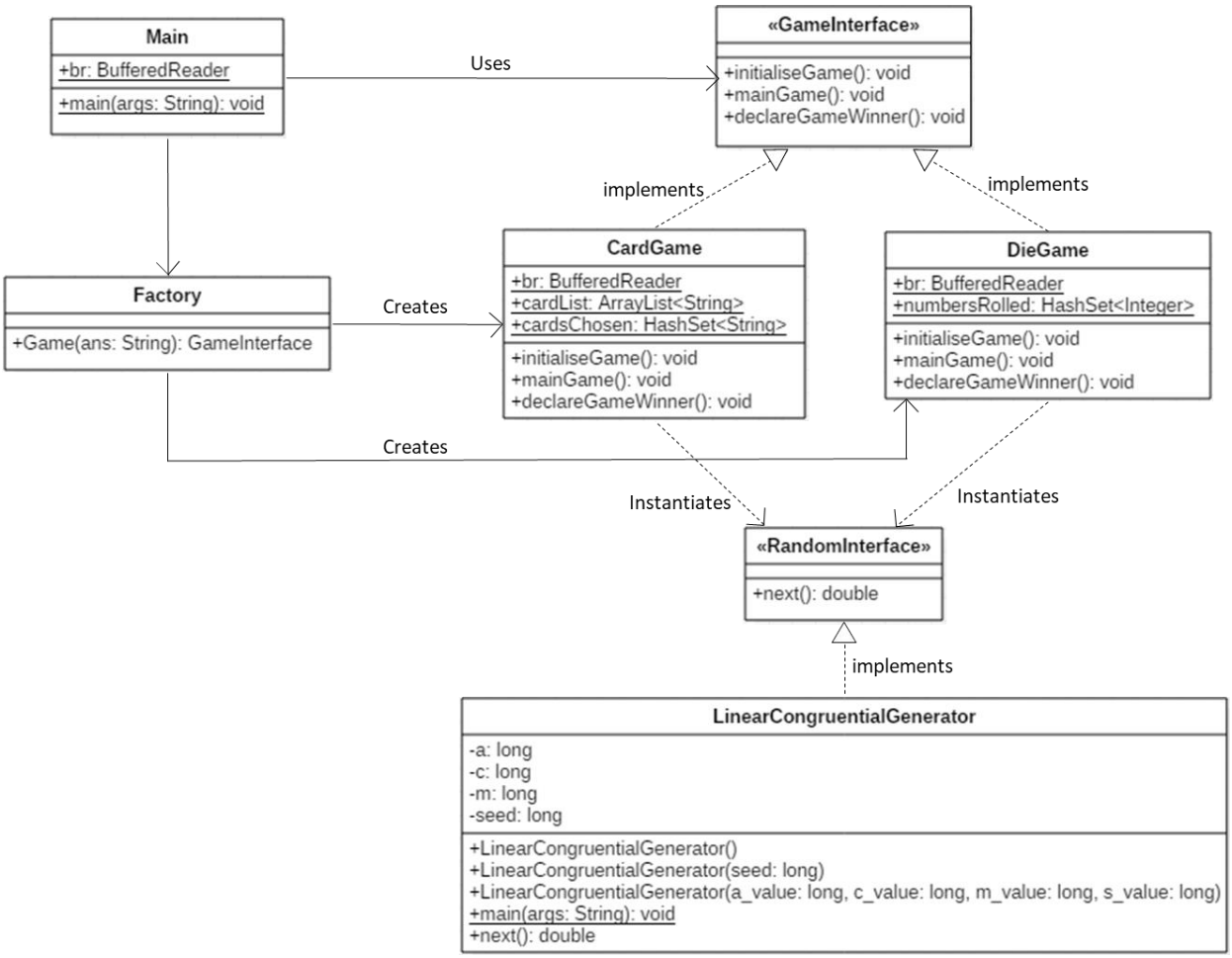


iii) The program contains 2 classes and an interface. The class "LinearCongruentialGenerator" implements the "RandomInterface" interface generating a random number normalized from 0 to 1. The "Game" class instantiates the value given from the LinearCongruentialGenerator and uses it on the methods that are used to play the different games. The current implementation of the program contains low cohesion since there are a lot of methods all doing several different things group into just 2 classes. Every class should be responsible for doing one thing so especially for the "Game" class there are 9 methods that should have been split into different classes in order to maximize cohesion. To optimize the program this class should be split into 3 different classes, "Main"," DieGame"," CardGame" focusing the methods in relation to the intention of the class.
On the other hand, the classes are loosely coupled since the "Game" class interacts with the "LinearCongruentialGenerator" class but depends on the "RandomInterface" Interface. This makes this is relatively good for the program since it makes it easier to change and maintain the code, but there is room for improvement. The classes are not dependent on the internal representation of the other, so they are not strongly connected.

Question 2:

i) By observing the structure of the program, we can conclude that it follows a Singleton pattern design restricting the instantiation of classes. The "Game" class contains several operations and attributes that can be respectively split into 3 different classes. Them being a "Main" class, a "DieGame" class and a "CardGame" class, containing their needed attributes and methods. As seen from Question 1(iii) we noted that the classes have low cohesion by having several non-related attributes and methods packed into 1 class. In order to improve our program, we need to establish a different design pattern. Firstly, by dividing the attributes and methods relating only to the intention of the class we increase the cohesion since they focus on only doing 1 thing. Secondly, we loosen the coupling by adding an interface which implements the methods of the "DieGame" and "CardGame" class and is being used directly from the "Main" class. This makes the loosens the dependency amongst those classes even more in relation to Question 1, improving the structure of our program. In order for the separation of the two game classes and the implementation of the other interface to work we need to apply a Factory pattern design. This will create an instance of the Game being played and will inherit the methods used from the connected interface.

ii) UML Diagram for Question2 representing the improved program I am going to implement for part (iii).



iii) My improved program implements the UML diagram representation of the used classes, interfaces and factory. It uses the “Main” class where the program is initialized calling the “Factory” where user input will decide which Factory Object which in this case being the game that will be created. Continuing it will use the “GameInterface” which contains the operations being created in each game class (“DieGame”, “CardGame”) telling the program which operations will be used. Inside the Card or Die game instances of the “RandomInterface” will be called being used by the operation “next()” which in turns implements the “LinearCongruentialGenerator” class creating a random number in normalized form from 0 to 1.

CardGame

Winning Game

```
Card (c) or Die (d) game? c
[6Hrts, JSpds, 5Dmnds, 2Clbs, 3Spds, 3Dmnds, 10Spds, ADmnds, 7Hrts, 2Spds
Hit <RETURN> to choose a card

You chose AClbs
Hit <RETURN> to choose a card

You chose 2Clbs
Cards chosen: [2Clbs, AClbs]
Remaining cards: [6Hrts, JSpds, 5Dmnds, 3Spds, 3Dmnds, 10Spds, ADmnds, 7H
Cards chosen: [2Clbs, AClbs]
You won!

Process finished with exit code 0
```

Losing Game

```
Card (c) or Die (d) game? c
[JDmnds, 7Dmnds, 4Hrts, ASpds, 5Spds, 2Dmnds, 4Dmnds, 3Dmnds, 5Clb
Hit <RETURN> to choose a card

You chose 8Spds
Hit <RETURN> to choose a card

You chose 10Dmnds
Cards chosen: [8Spds, 10Dmnds]
Remaining cards: [JDmnds, 7Dmnds, 4Hrts, ASpds, 5Spds, 2Dmnds, 4Dm
Cards chosen: [8Spds, 10Dmnds]
You lost!

Process finished with exit code 0
```

DieGame

Winning Game

```
Card (c) or Die (d) game? d
Hit <RETURN> to roll the die

You rolled 1
Hit <RETURN> to roll the die

You rolled 3
Numbers rolled: [1, 3]
You won!

Process finished with exit code 0
```

Losing Game

```
Card (c) or Die (d) game? d
Hit <RETURN> to roll the die

You rolled 2
Hit <RETURN> to roll the die

You rolled 4
Numbers rolled: [2, 4]
You lost!

Process finished with exit code 0
```

Wrong Inputs

```
Card (c) or Die (d) game? abc
Input not understood

Process finished with exit code 1
```

```
Card (c) or Die (d) game? cd
Input not understood

Process finished with exit code 1
```

```
Card (c) or Die (d) game? 123
Input not understood

Process finished with exit code 1
```

```
Card (c) or Die (d) game? c1
Input not understood

Process finished with exit code 1
```