Object-Oriented Applications: Coursework 1

C1646151 – Charles Read

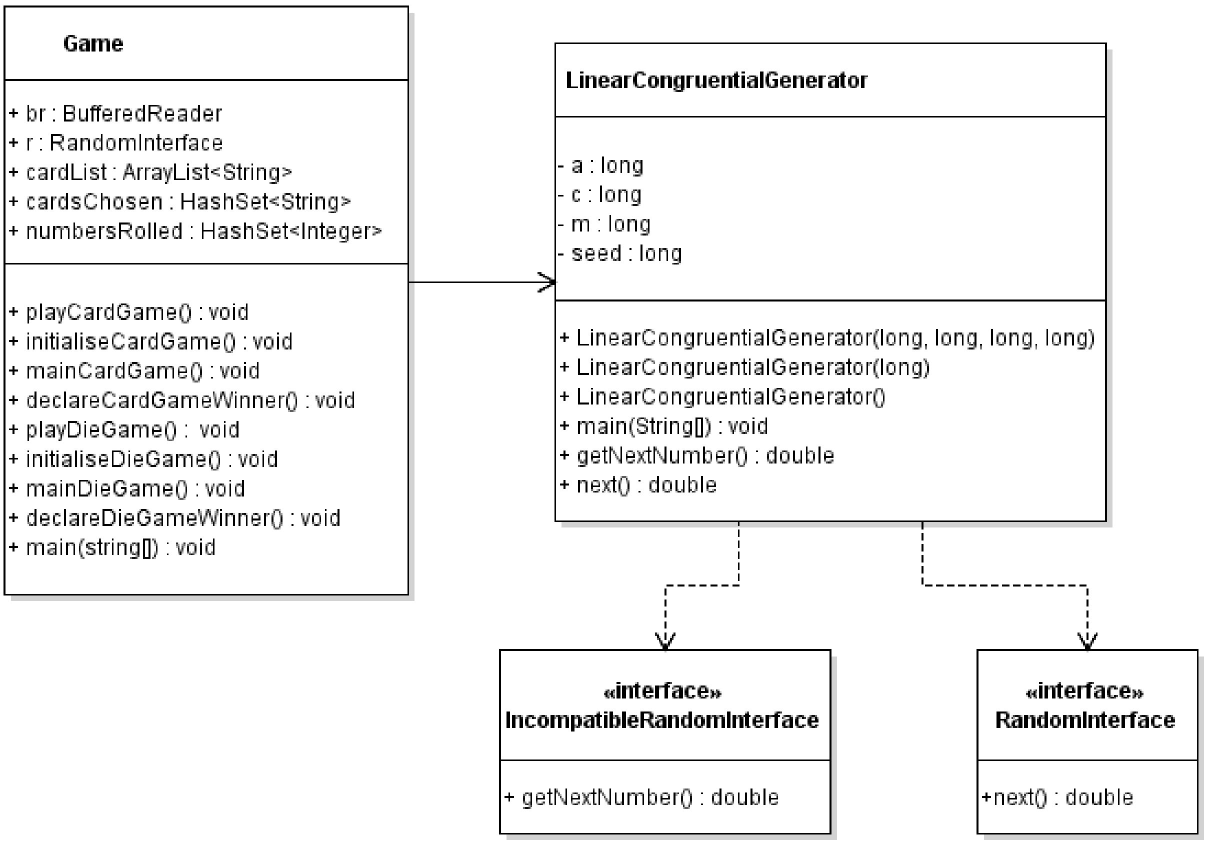
Question One i

"Fix" this problem by altering LinearCongruentialGenerator to implement RandomInterface.

See 'Part 1 Code' listings at the end of this PDF for changes.

Question one ii

Draw a UML class diagram representing the program which results from the above modifications.



Game
+ br : BufferedReader
+ r : RandomInterface
+ cardList : ArrayList<String>
+ cardsChosen : HashSet<String>
+ numbersRolled : HashSet<Integer>
+ playCardGame() : void
+ initialiseCardGame() : void
+ mainCardGame() : void
+ declareCardGameWinner() : void
+ playDieGame() : void
+ initialiseDieGame() : void
+ mainDieGame() : void
+ declareDieGameWinner() : void
+ main(string[]) : void
LinearCongruentialGenerator
- a : long
- c : long
- m : long
- seed : long
+ LinearCongruentialGenerator(long, long, long, long)
+ LinearCongruentialGenerator(long)
+ LinearCongruentialGenerator()
+ main(String[]) : void
+ getNextNumber() : double
+ next() : double
«interface»
IncompatibleRandomInterface
+ getNextNumber() : double
«interface»
RandomInterface
+next() : double

Question One iii

Brief Description

The program runs either a die game or a card game depending on the users input. The card game randomly selects a pair of cards for the user, shuffling after each withdrawal. If the player gets at least one ace of any suit, they win. In the die game, two 6-sided die are rolled at random. The user wins if at least one of the die rolled is a 1.

The program has a main class (game.java) that contains all the methods for running both the card and dice games. The 'LinearCongruentialGenerator.java' class generates pseudo-random numbers to be used by the game methods within game.java.

‘LinearCongruentialGenerator.java’ implements two interfaces; ‘RandomInterface.java’ defines a method for retrieving the next random number and ‘IncompatibleRandomInterface.java’, which also defines a method for retrieving the next random number.

The majority of the program is written in a procedural way, using static methods which isn’t an elegant way of programming in the object-oriented paradigm. The program also doesn’t take advantage of design patterns elements such as a factory and interfaces.

Question Two i

Identify suitable classes for an improved version of the program.

Game.java - This is the main class of the program that is ran when compiling.

GameGenerator.java - A Factory used in the program, this selects the game (Die or Card).

LinearCongruentialGenerator.java - Generates pseudo-random numbers to be used by the GameCard and GameDice classes.

GameInterface.java - An interface used in the program, used by GameGenerator and the rest of the program classes.

GameDice.java - A class that contains the methods related to the dice game.

GameCard.java - A class that contains the methods related to the card game.

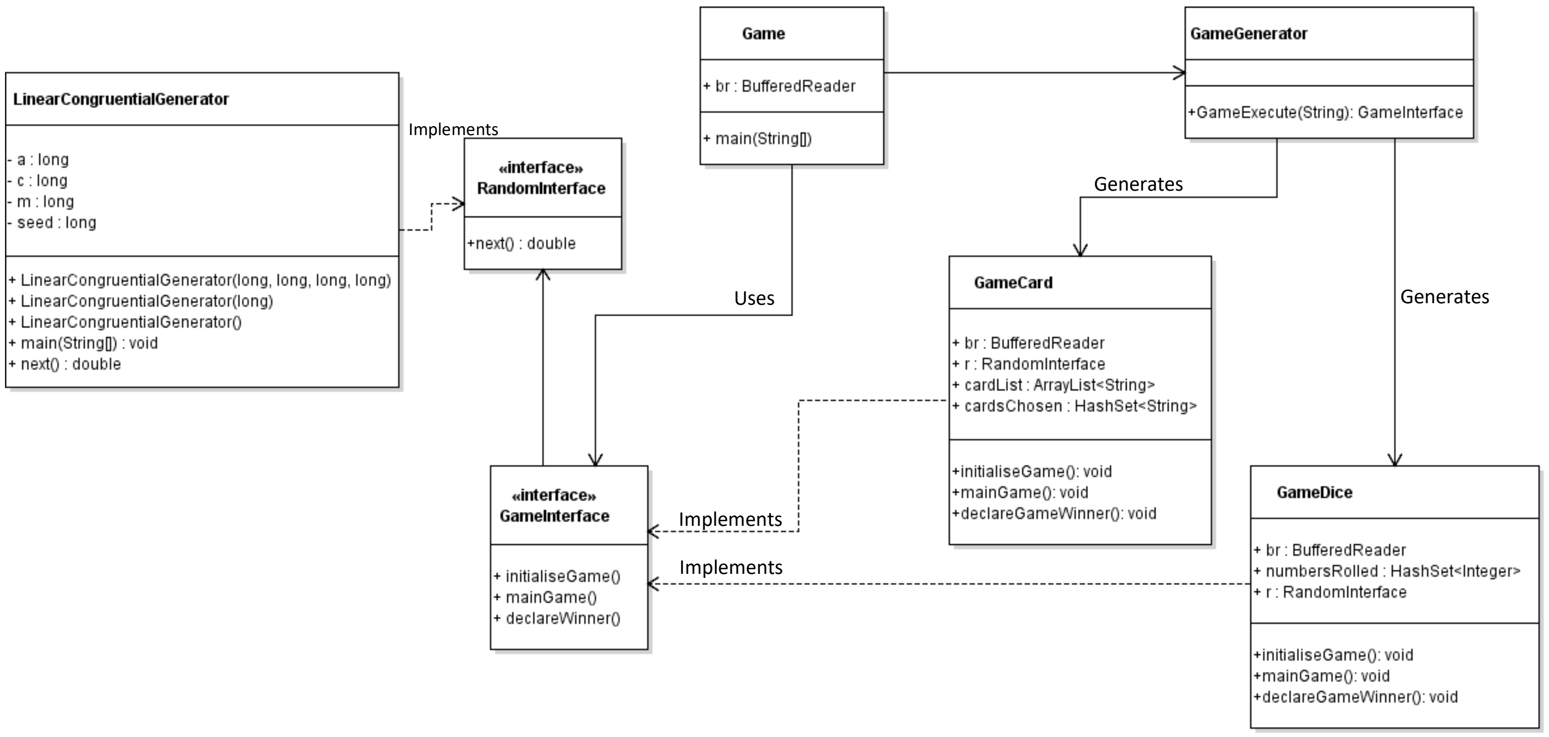
RandomInterface.java - An interface that defines a method for retrieving the next random number.

In this concept, to separate out the two game implantations, I have chosen to use the factory method. Using this factory allows more games to be added to the program later. The factory would interpret the user’s input and create a new instance of a game class (e.g. GameCard.java or GameDice.java), it would then use the GameInterface to perform the methods in the corresponding game class.

Using the factory design would give extensibility to the program as new games could be added to the program easily by adding the new game class and the new option to the factory. The factory design pattern would also offer ease of testing because of its modular structure.

Question Two ii

Draw a UML class diagram representing the improved program.



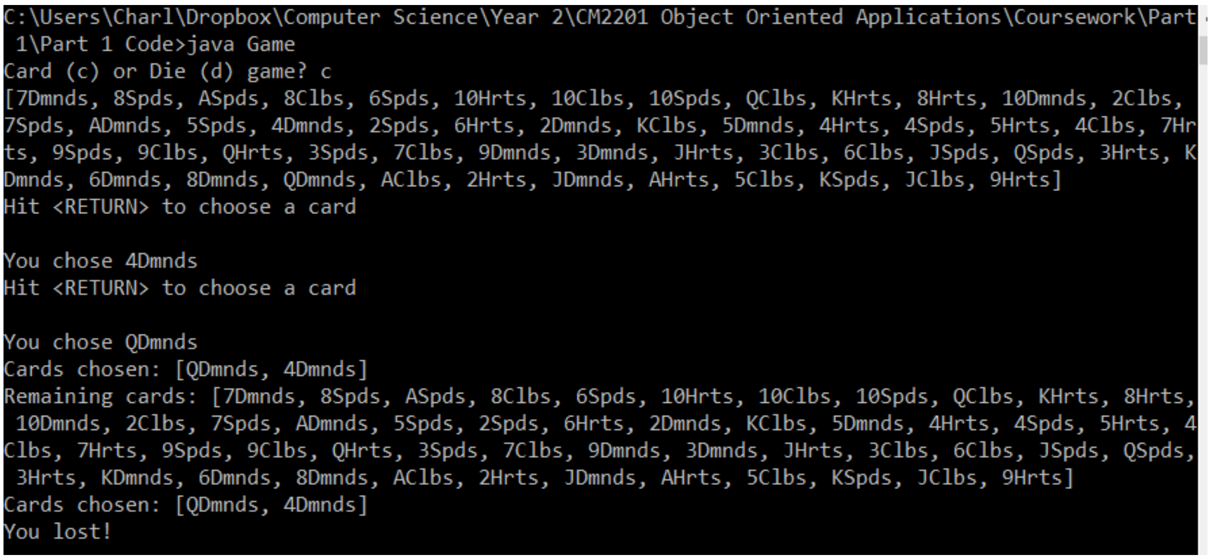
Question Two iii

Implement improved Program.

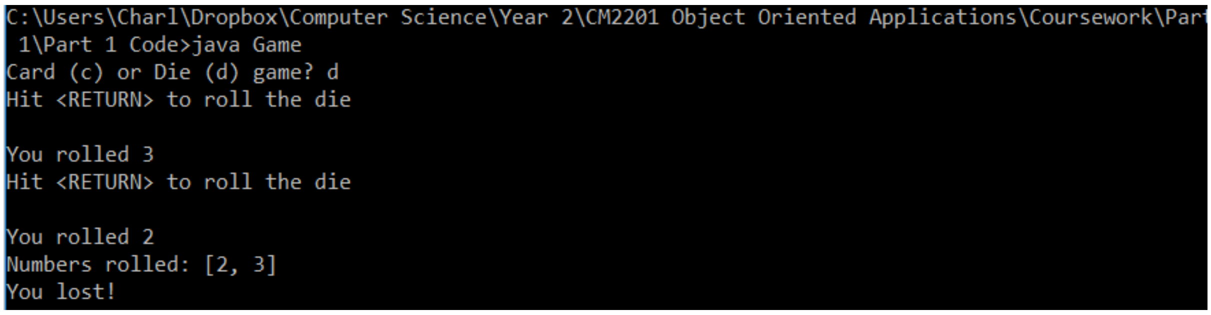
See ‘Part 2 Code’ listings at the end of this PDF.

Evidence

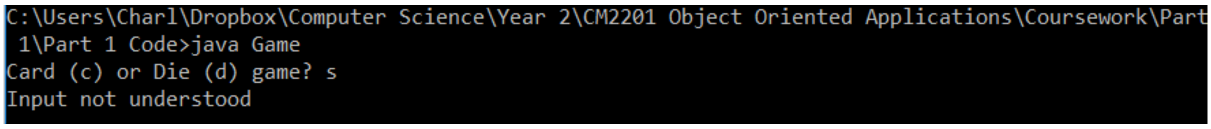
Question 1 Evidence



```
C:\Users\Charl\Dropbox\Computer Science\Year 2\CM2201 Object Oriented Applications\Coursework\Part
 1\Part 1 Code>java Game
Card (c) or Die (d) game? c
[7Dmnds, 8Spds, ASpds, 8Clbs, 6Spds, 10Hrts, 10Clbs, 10Spds, QClbs, KHrts, 8Hrts, 10Dmnds, 2Clbs,
7Spds, ADmnds, 5Spds, 4Dmnds, 2Spds, 6Hrts, 2Dmnds, KClbs, 5Dmnds, 4Hrts, 4Spds, 5Hrts, 4Clbs, 7Hr
ts, 9Spds, 9Clbs, QHrts, 3Spds, 7Clbs, 9Dmnds, 3Dmnds, JHrts, 3Clbs, 6Clbs, JSpds, QSpds, 3Hrts, K
Dmnds, 6Dmnds, 8Dmnds, QDmnds, AClbs, 2Hrts, JDmnds, AHrts, 5Clbs, KSpds, JClbs, 9Hrts]
Hit <RETURN> to choose a card

You chose 4Dmnds
Hit <RETURN> to choose a card

You chose QDmnds
Cards chosen: [QDmnds, 4Dmnds]
Remaining cards: [7Dmnds, 8Spds, ASpds, 8Clbs, 6Spds, 10Hrts, 10Clbs, 10Spds, QClbs, KHrts, 8Hrts,
 10Dmnds, 2Clbs, 7Spds, ADmnds, 5Spds, 2Spds, 6Hrts, 2Dmnds, KClbs, 5Dmnds, 4Hrts, 4Spds, 5Hrts, 4
Clbs, 7Hrts, 9Spds, 9Clbs, QHrts, 3Spds, 7Clbs, 9Dmnds, 3Dmnds, JHrts, 3Clbs, 6Clbs, JSpds, QSpds,
 3Hrts, KDmnds, 6Dmnds, 8Dmnds, AClbs, 2Hrts, JDmnds, AHrts, 5Clbs, KSpds, JClbs, 9Hrts]
Cards chosen: [QDmnds, 4Dmnds]
You lost!
```

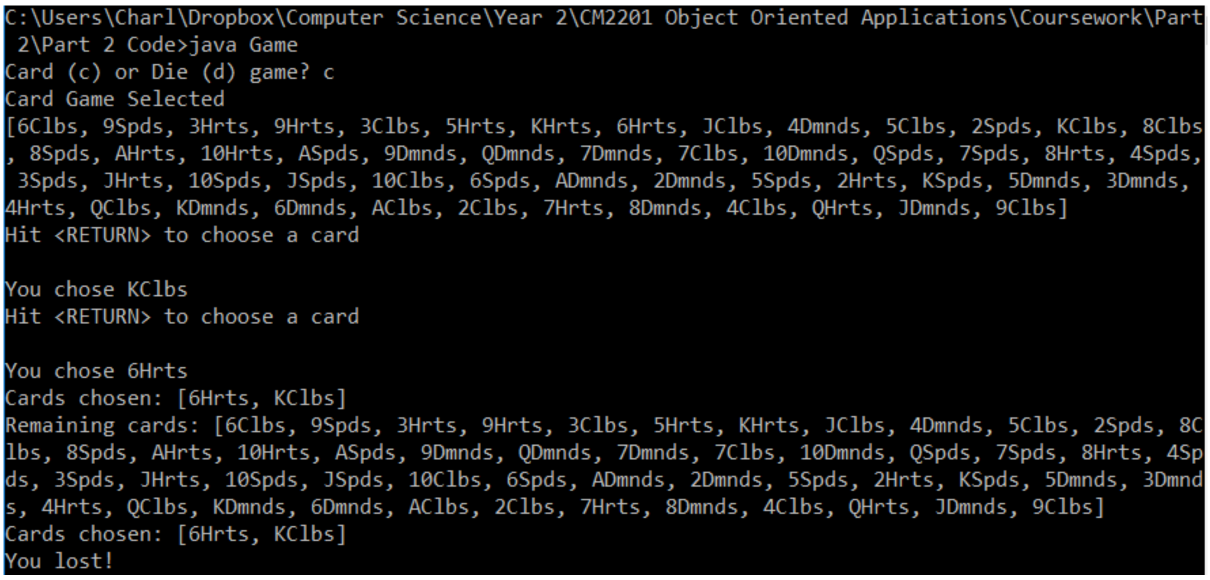


```
C:\Users\Charl\Dropbox\Computer Science\Year 2\CM2201 Object Oriented Applications\Coursework\Part
 1\Part 1 Code>java Game
Card (c) or Die (d) game? d
Hit <RETURN> to roll the die

You rolled 3
Hit <RETURN> to roll the die

You rolled 2
Numbers rolled: [2, 3]
You lost!
```

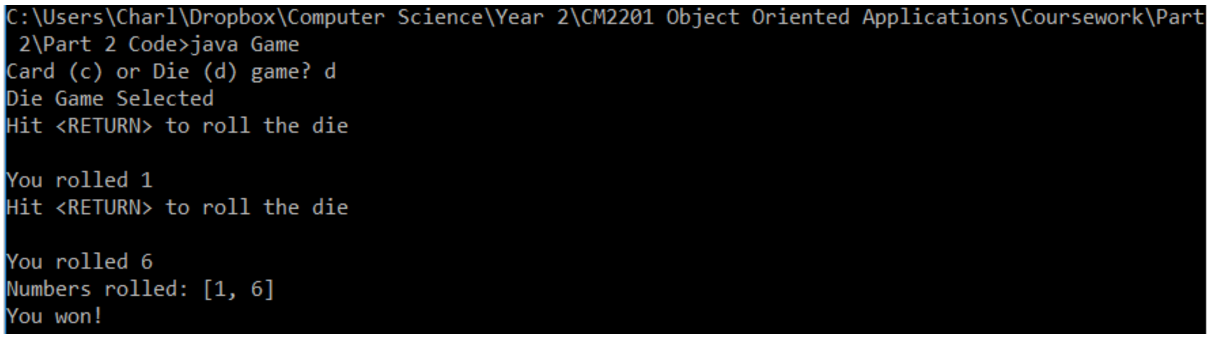


```
C:\Users\Charl\Dropbox\Computer Science\Year 2\CM2201 Object Oriented Applications\Coursework\Part
 1\Part 1 Code>java Game
Card (c) or Die (d) game? s
Input not understood
```

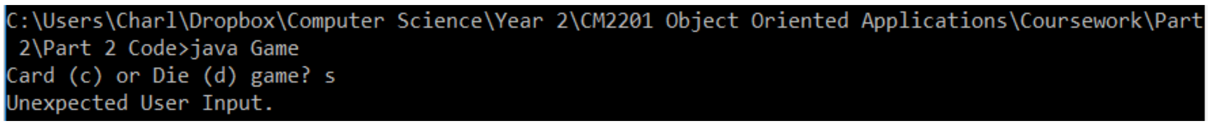
Question 2 Evidence



```
C:\Users\Charl\Dropbox\Computer Science\Year 2\CM2201 Object Oriented Applications\Coursework\Part
 2\Part 2 Code>java Game
Card (c) or Die (d) game? c
Card Game Selected
[6Clbs, 9Spds, 3Hrts, 9Hrts, 3Clbs, 5Hrts, KHrts, 6Hrts, JClbs, 4Dmnds, 5Clbs, 2Spds, KClbs, 8Clbs
, 8Spds, AHrts, 10Hrts, ASpds, 9Dmnds, QDmnds, 7Dmnds, 7Clbs, 10Dmnds, QSpds, 7Spds, 8Hrts, 4Spds,
 3Spds, JHrts, 10Spds, JSpds, 10Clbs, 6Spds, ADmnds, 2Dmnds, 5Spds, 2Hrts, KSpds, 5Dmnds, 3Dmnds,
4Hrts, QClbs, KDmnds, 6Dmnds, AClbs, 2Clbs, 7Hrts, 8Dmnds, 4Clbs, QHrts, JDmnds, 9Clbs]
Hit <RETURN> to choose a card

You chose KClbs
Hit <RETURN> to choose a card

You chose 6Hrts
Cards chosen: [6Hrts, KClbs]
Remaining cards: [6Clbs, 9Spds, 3Hrts, 9Hrts, 3Clbs, 5Hrts, KHrts, JClbs, 4Dmnds, 5Clbs, 2Spds, 8C
lbs, 8Spds, AHrts, 10Hrts, ASpds, 9Dmnds, QDmnds, 7Dmnds, 7Clbs, 10Dmnds, QSpds, 7Spds, 8Hrts, 4Sp
ds, 3Spds, JHrts, 10Spds, JSpds, 10Clbs, 6Spds, ADmnds, 2Dmnds, 5Spds, 2Hrts, KSpds, 5Dmnds, 3Dmnd
s, 4Hrts, QClbs, KDmnds, 6Dmnds, AClbs, 2Clbs, 7Hrts, 8Dmnds, 4Clbs, QHrts, JDmnds, 9Clbs]
Cards chosen: [6Hrts, KClbs]
You lost!
```



```
C:\Users\Charl\Dropbox\Computer Science\Year 2\CM2201 Object Oriented Applications\Coursework\Part
 2\Part 2 Code>java Game
Card (c) or Die (d) game? d
Die Game Selected
Hit <RETURN> to roll the die

You rolled 1
Hit <RETURN> to roll the die

You rolled 6
Numbers rolled: [1, 6]
You won!
```



```
C:\Users\Charl\Dropbox\Computer Science\Year 2\CM2201 Object Oriented Applications\Coursework\Part
 2\Part 2 Code>java Game
Card (c) or Die (d) game? s
Unexpected User Input.
```

Program Listings

Part 1 Code

Changes in Bold. – I only had changes in LinearCongruentialGenerator.java for part 1.

LinearCongruentialGenerator.java

public class LinearCongruentialGenerator implements IncompatibleRandomInterface**,**
RandomInterface {

```
// Generates pseudo-random numbers using:
// X(n+1) = (aX(n) + c) (mod m)
// for suitable a, c and m. The numbers are "normalised" to the range
// [0, 1) by computing X(n+1) / m.

  private long a, c, m, seed;
// Need to be long in order to hold typical values ...

  public LinearCongruentialGenerator(long a_value, long c_value, long m_value, long s_value) {
    a=a_value; c=c_value; m=m_value; seed=s_value;
  }

  public LinearCongruentialGenerator(long seed) {
  // Set a, c and m to values suggested in Press, Teukolsky, et al., "Numberical Recipies"
    this(1664525, 1013904223, 4294967296l, seed);
  // NB "l" on the end is the way that a long integer can be specified. The
  // smaller ones are type-cast silently to longs, but the large number is too
  // big to fit into an ordinary int, so needs to be defined explicitly
  }

  public LinearCongruentialGenerator() {
  // (Re-)set seed to an arbitrary value, having first constructed the object using
  // zero as the seed. The point is that we don't know what m is until after it has
  // been initialised.
```

```
 this(0);  seed=System.currentTimeMillis() % m;

 }

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

 // Since RandomInterface doesn't know about the instance variables defined in this
 // particular implementation, LinearCongruentialGenerator, we need to type-cast
 // in order to print out the parameters (primarily for "debugging" purposes).

  LinearCongruentialGenerator temp=(LinearCongruentialGenerator) r;
  System.out.println("a: " + temp.a + "  c: " + temp.c + "  m: " + temp.m + "  seed: " + temp.seed);

 }

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

  public double next(){
 return getNextNumber();
 }

}
```

Part 2 Code

Game.java is the main class that should be ran.

Game.java

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

public class Game {
	public static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

	public static void main(String[] args) {
		GameGenerator GameGen = new GameGenerator(); //Creates instance of
GameGenerator(my factory) called GameGen.

	try {
		System.out.print("Card (c) or Die (d) game? ");
		String GameAnswer = br.readLine(); //GameAnswer = C or D (depending on game).
		GameInterface GameExecute = GameGen.GameExecute(GameAnswer);  //Creates
instance of GameInterface called GameExecute.
```

```
                    GameExecute.initialiseGame(); //Calls methods from Interface instance.
                    GameExecute.mainGame();
                    GameExecute.declareGameWinner();
                    }

          catch (Exception e){
                              System.out.println("Unexpected User Input.");
                    }
          }
}
```

GameGenerator.java

```
public class GameGenerator { //GameGenerator is my factory class.
  public GameInterface GameExecute(String GameAnswer) throws Exception {

    //if statement to select the game type.
    if (GameAnswer.equals("c")){
             System.out.println("Card Game Selected");
             return new GameCard(); //creates instanse of GameCard class.
    }
    else if (GameAnswer.equals("d")){
             System.out.println("Die Game Selected");
             return new GameDice(); //creates instanse of GameDice class.
    }

    else throw new Exception();
  }
}
```

GameInterface.java

```
public interface GameInterface { //Interface of the program, calls methods for the Game classes
(GameDice, GameCard).
          void initialiseGame();
          void mainGame();
          void declareGameWinner();
}
```

GameCard.java

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

//class implements GameInterface for initialisation, mainGame and declareWinner.
public class GameCard implements GameInterface {

  public static BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
  public static RandomInterface r = new LinearCongruentialGenerator();

  // Variables used in GameCard class.
  public static ArrayList<String> cardList;
  public static HashSet<String> cardsChosen = new HashSet<String>();


  public void initialiseGame() {
    // The initialisation phase:

    // Create a list of cards ... and shuffle them
    String cards[]={"AHrts", "2Hrts", "3Hrts", "4Hrts", "5Hrts", "6Hrts",
                "7Hrts", "8Hrts", "9Hrts", "10Hrts", "JHrts",
                "QHrts", "KHrts",
                "ADmnds", "2Dmnds", "3Dmnds", "4Dmnds", "5Dmnds",
                "6Dmnds", "7Dmnds", "8Dmnds", "9Dmnds", "10Dmnds",
                "JDmnds", "QDmnds", "KDmnds",
                "ASpds", "2Spds", "3Spds", "4Spds", "5Spds", "6Spds",
                "7Spds", "8Spds", "9Spds", "10Spds", "JSpds",
                "QSpds", "KSpds",
                "AClbs", "2Clbs", "3Clbs", "4Clbs", "5Clbs", "6Clbs",
                "7Clbs", "8Clbs", "9Clbs", "10Clbs", "JClbs",
                "QClbs", "KClbs"};
    cardList=new ArrayList<String> (Arrays.asList(cards));
    // Taking advantage of "generics" to tell the compiler all the elements will be Strings

    // Shuffle them
    for (int i=0; i<100; i++) {
      // choose two random cards at random and swap them, 100 times
      int firstIndex=((int) (r.next() * 52));
      int secondIndex=((int) (r.next() * 52));
      String temp=(String) cardList.get(firstIndex);
      cardList.set(firstIndex, cardList.get(secondIndex));
      cardList.set(secondIndex, temp);
    }

    // Print out the result
    System.out.println(cardList);
  }

  public void mainGame() {
    // The main game:
    // Let user select two cards from the pack
    for (int i=0; i<2; i++) {
      System.out.println("Hit <RETURN> to choose a card");

     try {
      br.readLine();
     }
     catch(IOException e){
      System.out.println(e);
     }

      int cardChoice = (int) (r.next() * cardList.size());
      System.out.println("You chose " + cardList.get(cardChoice));
      cardsChosen.add(cardList.remove(cardChoice));
    }

    // Display the cards chosen and remaining cards
    System.out.println("Cards chosen: " + cardsChosen);
    System.out.println("Remaining cards: " + cardList);
  }

  public void declareGameWinner() {
    // Declare the winner:
    // User wins if one of them is an Ace
    System.out.println("Cards chosen: " + cardsChosen);
    if (cardsChosen.contains("AHrts") || cardsChosen.contains("ADmnds") ||
      cardsChosen.contains("ASpds") || cardsChosen.contains("AClbs")) {
     System.out.println("You won!");
    }
    else System.out.println("You lost!");
  }
}
```

GameDice.java

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

//class implements GameInterface for initialisation, mainGame and declareWinner.
public class GameDice implements GameInterface {

  public static RandomInterface r=new LinearCongruentialGenerator();

//Variables used in GameDice class.
  public static HashSet<Integer> numbersRolled=new HashSet<Integer>();
  public static BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

  public void initialiseGame() {
    // The initialization phase:
    // GameDice doesn't have an initialiseGame method.
  }
  public void mainGame() {
    // The main game:

    // Let the user roll the die twice
    for (int i=0; i<2; i++) {
      System.out.println("Hit <RETURN> to roll the die");
      try {
        br.readLine();
      }
      catch(IOException e){
        System.out.println(e);
      }
      int dieRoll=(int)(r.next() * 6) + 1;

      System.out.println("You rolled " + dieRoll);
      numbersRolled.add(new Integer(dieRoll));
    }

    // Display the numbers rolled
    System.out.println("Numbers rolled: " + numbersRolled);
  }

  public void declareGameWinner() {
    // Declare the winner:

    // User wins if at least one of the die rolls is a 1
    if (numbersRolled.contains(new Integer(1))) {
      System.out.println("You won!");
    }
    else System.out.println("You lost!");
  }
}
```

LinearCongruentialGenerator.java

```
public class LinearCongruentialGenerator implements RandomInterface {
// Generates pseudo-random numbers using:
// X(n+1) = (aX(n) + c) (mod m)
// for suitable a, c and m. The numbers are "normalised" to the range
// [0, 1) by computing X(n+1) / m.

  private long a, c, m, seed;
// Need to be long in order to hold typical values ...

  public LinearCongruentialGenerator(long a_value, long c_value, long m_value, long s_value) {
    a=a_value; c=c_value; m=m_value; seed=s_value;
  }

  public LinearCongruentialGenerator(long seed) {
  // Set a, c and m to values suggested in Press, Teukolsky, et al., "Numberical Recipies"
```

```
   this(1664525, 1013904223, 4294967296l, seed);
  // NB "l" on the end is the way that a long integer can be specified. The
  // smaller ones are type-cast silently to longs, but the large number is too
  // big to fit into an ordinary int, so needs to be defined explicitly
  }

  public LinearCongruentialGenerator() {
  // (Re-)set seed to an arbitrary value, having first constructed the object using
  // zero as the seed. The point is that we don't know what m is until after it has
  // been initialised.

  this(0);  seed=System.currentTimeMillis() % m;

  }

  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());

  // Since RandomInterface doesn't know about the instance variables defined in this
  // particular implementation, LinearCongruentialGenerator, we need to type-cast
  // in order to print out the parameters (primarily for "debugging" purposes).

    LinearCongruentialGenerator temp=(LinearCongruentialGenerator) r;
    System.out.println("a: " + temp.a + "  c: " + temp.c + "  m: " + temp.m + "  seed: " + temp.seed);

  }

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

RandomInterface.java

```
public interface RandomInterface {
// Simply defines a method for retrieving the next random number
  public double next();
}
```