Day 2: Introduction to Inheritance, Overriding, and Overloading

The Randomizer interface:

```
01 public interface Randomizer {
02   // Getters
03   public int getPossibleOutcomes();
04   public int getCurrentValue();
05   public String getCurrentFace();
06
07   // Setters (or Mutators)
08   public void randomize();
09 }
```

The Coin Randomizer Class:

```
01 // Class Name and Interface(s) Implemented by the Class
02 public class Coin implements Randomizer {
03 // Instance Variables
04   private boolean isHeads;
05
06 // Constructors
07   public Coin() {
08     isHeads=Math.random()<.5;
09   }
10 // Getters
11   public int getPossibleOutcomes() {
12     return 2;
13   }
14   public int getCurrentValue() {
15     if (isHeads==true) return 1;
16     return 0;
17   }
18   public String getCurrentFace() {
19     if (isHeads==true) return "Heads";
20     return "Tails";
21   }
22   // Overriding an inherited method from the Object Class
23   public String toString() {
24     return "The coin is showing "+getCurrentFace();
25   }
26 // Setter(s) or Mutator
27   public void randomize() {
28     isHeads=Math.random()<.5;
29   }
30 }
```

The Coin Runner Class:

```
01 public class CoinRunner {
02   public static void main(String[] args) {
03     Coin myCoin = new Coin(); // Instantiate a Coin Object
04     System.out.println("Initial Value="+myCoin.getCurrentFace());
05     int countHeads=0;
06     for (int i=0; i<10; i++) {
07       myCoin.randomize();
08       if (myCoin.getCurrentValue()==1) {
09         countHeads++;
10       }
11       System.out.println(myCoin);
12     }
13     System.out.println("Heads="+countHeads);
14   }
15 }
```

The D6 Class:

```
01 public class D6 implements Randomizer {
02   // Instance Variables
03   private int sideUp;
04   // Constructor
05   public D6() {// Default (a.k.a. No Parameters Constructor)
06     randomize();
07   }
08   // Getters
09   public int getPossibleOutcomes() {
10     return 6;
11   }
12   public int getCurrentValue() {
13     return sideUp;
14   }
15   public String getCurrentFace() {
16     return ""+sideUp;
17   }
18   // Overriding Inherited Method from the Object Class
19   public String toString() {
20     return "d6="+getCurrentFace();
21   }
22   // Setters (or Mutators)
23   public void randomize() {
24     // Cast double into an int
25     sideUp=(int)(Math.random()*6)+1;
26   }
27 }
```

The DiceRunner class (Day 2, Parts I & II):

```
01 public class DiceRunner {
02    public static void main(String[] args) {
03       D6 die=new D6();
04       System.out.println("Initial Value="+die.getCurrentFace());
05       /* Roll the die 10 times */
06       for (int i=0; i<10; i++) {
07          die.randomize();
08          System.out.println("Roll #"+(i+1)+", "+die);
09       }
10    }
11 }
```

The DiceBagRunner class (Day 2, Part III):

```
01 public class DiceBagRunner {
02    public static void main(String[] args) {
03       D6[] dice=new D6[3];
04       dice[0]=new D6();
05       dice[1]=new D6();
06       dice[2]=new D6();
07       for (int rollNum=1; rollNum<=10; rollNum++) {
08          System.out.print("Roll #"+rollNum+": ");
09          int diceTotal=0;
10          for (int dieIndex=0; dieIndex<dice.length; dieIndex++) {
11             D6 die=dice[dieIndex];
12             die.randomize();
13             diceTotal=diceTotal+die.getCurrentValue();
14             System.out.print(die+", ");
15          }
16          System.out.println(" total="+diceTotal);
17       }
18    }
19 }
```

The PolyhedralDie class:

```
01 public class PolyhedralDie implements Randomizer {
02    // Instance Variables
03    private int numberOfSides;
04    private int sideUp;
05
06    // Constructor Methods
07    public PolyhedralDie() { // Default Constructor
08       numberOfSides=6;
09       randomize();
10    }
11    // Overloaded Constructor
12    public PolyhedralDie(int setNumberOfSides) {
13       numberOfSides=setNumberOfSides;
14       randomize();
15    }
16    // Getter Methods
17    public int getPossibleOutcomes() {
18       return numberOfSides;
19    }
20    public int getCurrentValue() {
21       return sideUp;
22    }
23    public String getCurrentFace() {
24       return ""+sideUp;
25    }
26    // Overridden Inherited toString() method from the Object class
27    public String toString() {
28       return "d"+getPossibleOutcomes()+"="+getCurrentFace();
29    }
30    // Setter Methods (or Mutator)
31    public void randomize() {
32       sideUp=(int)(Math.random()*numberOfSides)+1;
33    }
34 }
```

The DiceBagRunner class (Day 2, Part IV):

```
01 public class DiceBagRunner {
02    public static void main(String[] args) {
03       // Creating the dice array filled with null values
04       PolyhedralDie[] dice=new PolyhedralDie[3];
05       // Instantiating a Die in the dice array with the Default Constructor
06       dice[0]=new PolyhedralDie();
07       // Instantiating Dice in the dice array with the Overloaded Constructor
08       dice[1]=new PolyhedralDie(12);
09       dice[2]=new PolyhedralDie(20);
10       for (int rollNum=1; rollNum<=10; rollNum++) {
11          System.out.print("Roll #"+rollNum+": ");
12          int diceTotal=0;
13          for (int dieIndex=0; dieIndex<dice.length; dieIndex++) {
14             PolyhedralDie die=dice[dieIndex];
15             die.randomize();
16             diceTotal=diceTotal+die.getCurrentValue();
17             System.out.print(die+", ");
18          }
19          System.out.println(" total="+diceTotal);
20       }
21    }
22 }
```