Day 4: Abstract Classes and Extending Classes

The Randomizer interface (from Day 1): The Interface For the Coin, D6, and PolyhedralDie

```
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 (from Day 2): The Coin From the Previous Lesson

```
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 D6 Class (from Day 2): The Six Sided Die From the Previous Lesson

```
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 PolyhedralDie class (from Day 2): The PolyhedralDie From the Previous Lesson

```
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 LabeledPolyhedralDie class (Day 3, Part III) – Extending An Existing Class

```
01 public class LabeledPolyhedralDie extends PolyhedralDie {
02    // Instance Variables
03    private String[] sideLabels;
04
05    // Constructor Methods
06    public LabeledPolyhedralDie() {
07       super();
08       sideLabels=new String[getPossibleOutcomes()];
09       for (int i=0; i<sideLabels.length; i++) {
10          sideLabels[i]=""+i;
11       }
12    }
13    public LabeledPolyhedralDie(String[] setSideLabels) {
14       super(setSideLabels.length);
15       sideLabels=new String[getPossibleOutcomes()];
16       for (int i=0; i<sideLabels.length; i++) {
17          sideLabels[i]=setSideLabels[i];
18       }
19    }
20
21    // Getter Methods
22    public String getCurrentFace() {
23       return sideLabels[getCurrentValue()-1];
24    }
25    public String toString() {
26       return super.toString()+"("+getCurrentValue()+")";
27    }
28 }
```

The DiceBagRunner class (from Day 3, Part III) – Using our old classes

```
01 public class DiceBagRunner {
02    public static void main(String[] args) {
03       /* Creating the dice array filled with null values using the
04       *  interface as type rather than a specific class!
05       */
06       Randomizer[] dice=new Randomizer[3];
07       // Instantiating a new labeled die using overloaded constructor!
08       String[] sides={"Side 1","Side B","Side ?","Side !","Side Alpha"};
09       dice[0]=new LabeledDie(sides);
10       // Instantiate a LabeledPolyhedralDie with default constructor
11       dice[1]=new LabeledPolyhedralDie();
12       // Instantiate a LabeledPolyhedralDie with overloaded constructor
13       dice[2]=new LabeledPolyhedralDie(sides);
14       for (int rollNum=1; rollNum<=10; rollNum++) {
15          System.out.print("Roll #"+rollNum+": ");
16          int diceTotal=0;
17          for (int dieIndex=0; dieIndex<dice.length; dieIndex++) {
18             Randomizer die=dice[dieIndex];
19             die.randomize();
20             diceTotal=diceTotal+die.getCurrentValue();
21             System.out.print(die+", ");
22          }
23          System.out.println(" total="+diceTotal);
24       }
25    }
26 }
```

The AbstractRandomizer class (Day 4, Part I) – Creating an Abstract Randomizer

```
01 public abstract class AbstractRandomizer implements Randomizer {
02    private int sideUp;
03
04    // Getters
05    abstract public int getPossibleOutcomes();
06    public int getCurrentValue() {
07       return sideUp;
08    }
09    abstract public String getCurrentFace();
10
11    // Setters (or Mutators)
12    public void randomize() {
13       sideUp=(int)(Math.random()*getPossibleOutcomes())+1;
14    }
15 }
```

Re-writing the Coin class (Day 4, Part I) – Extending AbstractRandomizer to create a better Coin

```
01 public class Coin extends AbstractRandomizer {
02    public Coin() {
03       randomize();
04    }
05    public int getPossibleOutcomes() {
06       return 2;
07    }
08    public String getCurrentFace() {
09       if (getCurrentValue()==1) return "Tails";
10       return "Heads";
11    }
12    public String toString() {
13       return "The coin is showing "+getCurrentFace();
14    }
15 }
```

The DiceBagRunner class (Day 4, Parts I-III) – A Bigger Bag of Dice

```
01 public class DiceBagRunner {
02   public static void main(String[] args) {
03     // Dice array filled with null values of type Randomizer
04     Randomizer[] dice=new Randomizer[5];
05     // An array for storing the labels we will use for our dice
06     String[] sides={"Side 1","Side B","Side ?","Side !","Side Alpha"};
07     // Instantiating some Randomizers...
08     dice[0]=new Coin();
09     dice[1]=new D6();
10     dice[2]=new PolyhedralDie();
11     dice[2]=new PolyhedralDie(12);
12     dice[3]=new LabeledPolyhedralDie();
13     dice[4]=new LabeledPolyhedralDie(sides);
14     System.out.println("Initial Values:");
15     int diceTotal=0;
16     for (int dieIndex=0; dieIndex<dice.length; dieIndex++) {
17       Randomizer die=dice[dieIndex];
18       System.out.print(die+", ");
19     }
20     System.out.println(" total="+diceTotal);
21
22     for (int rollNum=1; rollNum<=10; rollNum++) {
23       System.out.print("Roll #"+rollNum+": ");
24       diceTotal=0;
25       for (int dieIndex=0; dieIndex<dice.length; dieIndex++) {
26         Randomizer die=dice[dieIndex];
27         die.randomize();
28         diceTotal=diceTotal+die.getCurrentValue();
29         System.out.print(die+", ");
30       }
31       System.out.println(" total="+diceTotal);
32     }
33   }
34 }
```

Re-writing the D6 class (Day 4, Part II) – Extending AbstractRandomizer to create a better D6

```
01 public class D6 extends AbstractRandomizer {
02    public D6() {
03       randomize();
04    }
05    public int getPossibleOutcomes() {
06       return 6;
07    }
08    public String getCurrentFace() {
09       return Integer.toString(getCurrentValue());
10    }
11    public String toString() {
12       return "d6="+getCurrentFace();
13    }
14 }
```

Re-writing the PolyhedralDie class (Day 4, Part III) – Making a better PolyhedralDie

```
01 public class PolyhedralDie extends AbstractRandomizer {
02    // Instance Variables
03    private int numSides;
04
05    // Constructor Methods
06    public PolyhedralDie() {
07       numSides=6;
08       randomize();
09    }
10    public PolyhedralDie(int setNumSides) {
11       numSides=setNumSides;
12       randomize();
13    }
14    // Getter Methods
15    public int getPossibleOutcomes() {
16       return numSides;
17    }
18    public String getCurrentFace() {
19       return Integer.toString(getCurrentValue());
20    }
21    public String toString() {
22       return "d"+numSides+"="+getCurrentFace();
23    }
24 }
```