DAY 3: POLYMORPHISM AND EXTENDING OBJECTS

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 DiceBagRunner class (from Day 2, Part IV): Our DiceBagRunner From the Previous Lesson

```
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 }
```

The DiceBagRunner (Day 3, Part I) – Mixing Different Randomizers In A Single Array

```
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 the type rather than a specific class!
05       */
06       Randomizer[] dice=new Randomizer[3];
07       // Instantiating a new D6 in the dice array
08       dice[0]=new D6();
09       // Instantiating a new PolyhedralDie in the dice array
10       dice[1]=new PolyhedralDie(20);
11       // Instantiating a new Coin in the dice array!
12       dice[2]=new Coin();
13       for (int rollNum=1; rollNum<=10; rollNum++) {
14          System.out.print("Roll #"+rollNum+": ");
15          int diceTotal=0;
16          for (int dieIndex=0; dieIndex<dice.length; dieIndex++) {
17             /* This is allowed because all the objects in the array
18             * implement the Randomizer interface, therefore they
19             * are all Randomizers
20             */
21             Randomizer die=dice[dieIndex];
22             /* The correct randomize method will be called for each
23              * Randomizer due to Polymorphism, the ability of each
24              * class to define it's own version of a method through
25              * a common interface
26              */
27             die.randomize();
28             /* Polymorphism again allows each die to define it's own
29              * way of getting the current value
30              */
31             diceTotal=diceTotal+die.getCurrentValue();
32             /* And finally, Polymorphism allows each die to be
33              * converted to a String for output using the combined
34              * powers of Polymorphism and Overiding inherited
35              * methods
36              */
37             System.out.print(die+", ");
38          }
39          System.out.println(" total="+diceTotal);
40       }
41    }
42 }
```

The LabeledDie class (Day 3, Part II): Creating A Labeled Die From Scratch

```
01 public class LabeledDie implements Randomizer {
02    // Instance Variables
03    String[] faces;
04    int faceUpIndex;
05    public LabeledDie() {
06       faces=new String[6];
07       for (int i=0; i<faces.length; i++) {
08          this.faces[i]=Integer.toString(i+1);
09       }
10       randomize();
11    }
12    public LabeledDie(String[] setFaces) {
13       faces=new String[setFaces.length];
14       for (int i=0; i<setFaces.length; i++) {
15          faces[i]=setFaces[i];
16       }
17       randomize();
18    }
19    // Getters
20    public int getPossibleOutcomes() {
21       return faces.length;
22    }
23    public int getCurrentValue() {
24       return faceUpIndex+1;
25    }
26    public String getCurrentFace() {
27       return faces[faceUpIndex];
28    }
29    // Setters (or Mutators)
30    public void randomize() {
31       faceUpIndex=(int)(Math.random()*faces.length);
32    }
33    public String toString() {
34       String s="d"+faces.length+"="+getCurrentFace();
35       return s+"("+getCurrentValue()+")";
36    }
37 }
```

The DiceBagRunner class (Day 3, Part II) – Adding a Labeled Die

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

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 (Day 3, Part III) – Comparing The Two Versions Of Labeled Die

```
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 }
```