IT 179
Java Review

What is an object?

- A dog is an **object**.
- It has attributes
  - age, weight, coat color, breed, gender, etc.
- It has behaviors
  - walking, running, barking, etc.

What is an object?

- Short answer:
  - An object is entity that has both attributes and behaviors.
- Attributes = data
- Behaviors = methods

Classes

- A class provides the template of an object.
- Classes provide the blueprint
  - You can make as many objects as you want from that blueprint.
- The attributes of the object are what differentiate the objects.
- All objects have the same methods, but they operate on different data.

Objects are _____ of classes?

As a programmer, you design and implement classes _____ you create objects.

- before
- after
- before or after

Example


```
public class Employee
{

   //atrributes
   private Integer SSN;
   private String phone;
   private String name; //private other objects cannot directly access name

   //methods
   public void setSSN(Integer SSN) {
      this.SSN = SSN;
   }

   public Integer getSSN() {
      return this.SSN;
   }

   public void setPhone(String phone) {
      this.phone = phone;
   }

   public String getPhone() {
      return this.phone;
   }

   public void setName(String name) {
      this.name = name;
   }

   public String getName() {
      return this.name;
   }
```

Creating Object from a Class


```
public class EmployeeTest
{
   public static void main(String[] args) {

       Employee employee = new Employee();
       employee.setName("Foo Bar");
       employee.getName(); //returns "Foo Bar"
   }

}
```

Constructors

- Constructors are special methods that have the same name as the class.
- Constructor are used to **initialize the attributes** in a newly created object.
- You use constructors when you create objects

Example


```
public class Circle {

  private double radius;
  private double x; //x coordinate of the center
  private double y; //y coordinate of the center

  public Circle () {
    this.radius = 1.0;
    this.x = 0.0;
    this.y = 0.0;
  }

  public Circle (double radius) {
    this.radius = radius;
    this.x = 0.0;
    this.y = 0.0;
  }

  public Circle (double radius, double x, double y) {
    this.radius = radius;
    this.x = x;
    this.y = y;
  }

}
```

Overloading

- Overloading occurs when methods have:
  - the **same** name but
  - **different** parameter signatures

```
Circle c1 = new Circle(); //create a unit circle, centered at (0,0)
Circle c2 = new Circle(3); //create a circle with a radius of 3
                           //and centered at (0,0)

Circle c3 = new Circle(5, 5, 5); //createa circle with radisu 5 and
                                 //centered at (5,5)
```

Inheritance

- Create parent class
- Child class (sub class) "inherits" attributes and behaviors from parent (super class)
- There is an **is-a** relationship between subclasses and their super class.
- Examples
  - A Dog is-a Mammal
  - A Rectangle is-a Shape
  - A Circle is-a Shape

Overriding

- **Overriding** allows a child class to implement a method that is already implemented by one of its parent classes

Overriding Example

```
class Animal {
   public void move() {
      System.out.println("Animals can move");
   }
}

class Dog extends Animal {
   public void move() {
      System.out.println("Dogs can walk and run");
   }
}

public class TestDog {

   public static void main(String args[]) {
      Animal a = new Animal();   // Animal reference and object
      Animal b = new Dog();   // Animal reference but Dog object

      a.move();   // runs the method in Animal class
      b.move();   // runs the method in Dog class
   }
}
```

Arrays

- An array is a sequence of variables of the **same** data type.
- **Declare an array**

```
datatype [] arrayName;
```

- **Instantiate an array**

```
arrayName = new datatype[size];
```

Arrays - Examples

- Declaring arrays

```
double [] dailyTemps; // elements are doubles
String [] cdTracks; // elements are Strings
boolean [] answers; // elements are booleans
Auto [] cars; // elements are Auto references
int [] cs101, bio201; // two int arrays
```

Arrays - Examples

- Instantiating these arrays

```
dailyTemps = new double[365]; // 365 elements
cdTracks = new String[15]; // 15 elements
int numberOfQuestions = 30;
answers = new boolean[numberOfQuestions];
cars = new Auto[3]; // 3 elements
cs101 = new int[5]; // 5 elements
bio201 = new int[4]; // 4 elements
```

First Element of An Array


```
public class Cell-Bills
{
  public static void main( String [ ] args )
  {
     // declare and instantiate the array
     double [ ] cell-Bills = new double [6];

     // assign values to array elements
     cellBills[0] = 45.24;
     cellBills[1] = 54.67;
     cellBills[2] = 42.55;
     cellBills[3] = 44.61;
     cellBills[4] = 65.29;
     cellBills[5] = 49.75;

     System.out.println( "The first monthly cell bill is "
                         + cellBills[0] );
     System.out.println( "The last monthly cell bill is "
                         + cellBills[cellBills.length - 1] );
  }
}
```

Instantiating an Array of Objects


```
// instantiate array; all elements are null
Auto [] cars = new Auto[3];
// instantiate objects and assign to elements
Auto sportsCar = new Auto( "Miata", 100, 5.0 );
cars[0] = sportsCar;
cars[1] = new Auto( );
// cars[2] is still null
```

Operations on All Elements of an Array

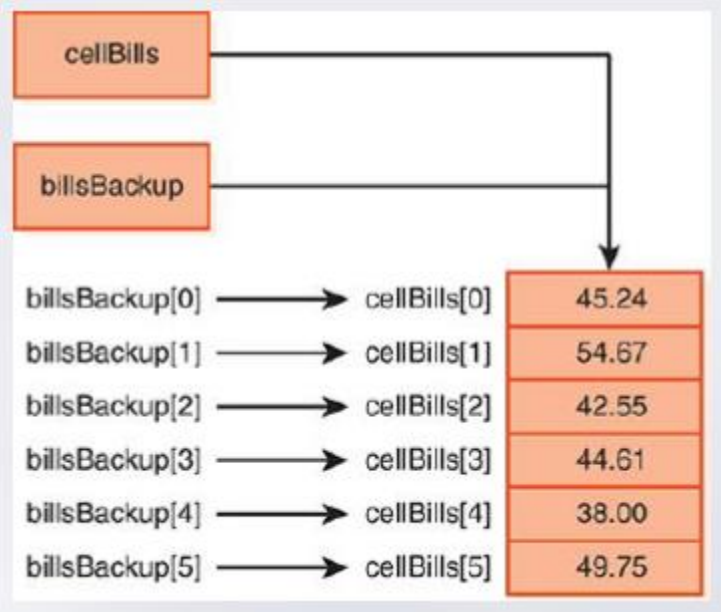

```
for ( int i = 0; i < cellBills.length; i++ )
{
    System.out.println( cellBills[i] );
}
```

Copying Arrays


```
double [] billsBackup = new double [6];
billsBackup = cellBills; // incorrect!
```

Copying Arrays

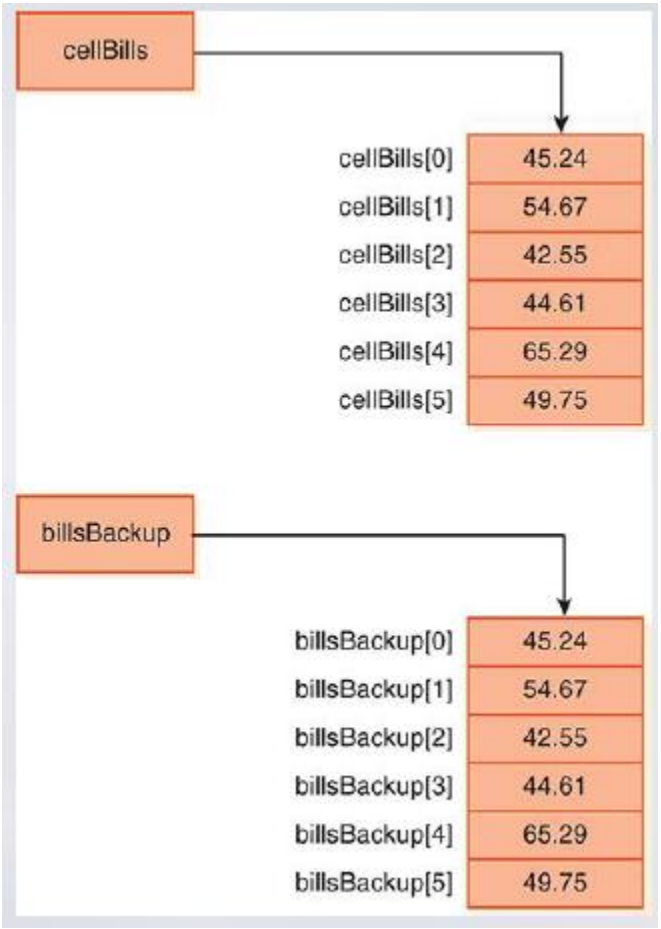
- billsBackup = cellBills;



Copying Arrays


```
for ( int i = 0; i < cellBills.length; i++ )
{
   billsBackup[i] = cellBills[i];
}
```

```
for ( int i = 0; i < cellBills.length; i++ )
{
   billsBackup[i] = cellBills[i];
}
```



The Switch Statement


```
switch ( expression )
{
    case constant1:
        // statement(s);
        break;    // optional
    case constant2:
        // statement(s);
        break;    // optional
      …
    default:      // optional
        statement(s);
        …
}
```

```
int day = 4;
switch (day) {
  case 6:
    System.out.println("Today is Saturday");
    break;
  case 7:
    System.out.println("Today is Sunday");
    break;
  default:
    System.out.println("Looking forward to the Weekend");
```

What does this print out?


```
int x = 1;
    switch(x) {
      case 1: System.out.println("Case One");
      case 3: System.out.println("Case Three");
      default: System.out.println("Case Default");
    }
```

Case One
Case Three
Case Default

What does this print out?


```
int x = 1;
    switch(x) {
      case 1: System.out.println("Case One");break;
      case 3: System.out.println("Case Three");break;
      default: System.out.println("Case Default");
    }
```

Case One

do {..} while


```
int x = 1;
    do {
        System.out.println("x = " + x++ );
    }
    while (x < 5);
```

x = 1
x = 2
x = 3
x = 4

do {..} while


```
int x = 1;
    do {
        System.out.println("x = " + x++ );
    }
    while (--x < 5);
```

x = 1
x = 1
x = 1
x = 1
.
.
.
.

What does this print out?


```
int[] array = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
    int i = 0;
    do {
        System.out.println(--array[++i]);
    }
    while (i < array.length );
```

19
29
39
49
59
69
79
89
99

What does this print out?


```
int[] array = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100};
    int i = 0;
    do {
        System.out.println(--array[i++]);
    }
    while (i < array.length );
```

9
19
29
39
49
59
69
79
89
99

Entrance Exam

- Part 1 – 22 questions, 61 points
- Part 2 – 39 points
- Due: Friday, @ 11:55pm