Overview

- Classes – w/ static methods
- Classes – w/ non-static methods
- Access/Visibility Modifiers
- Instance Variables
- Methods
- Constructors
- Static vs. Non-static
We have used already used other classes that have static methods:

- Integer - parseInt
- Double - parseDouble

These are classes that do not need be instantiated (that use the keyword “new”)

Let's take a look at what the Integer class may look like:

```java
public class Integer {
    public static int parseInt(String s) {
        int val = 0;
        for (int i = 0; i < s.length(); i++) {
            int lVal = (int) (s.charAt(i) - '0');
            val = val * 10 + lVal;
        }
        return val;
    }
}
```
Classes

Classes with non-static methods are classes that need to be instantiated as an object.

Ok what does that mean?
This means that we need to create a variable of the class type.

We have seen examples of this with other classes:
- BufferedReader r = new BufferedReader(new InputStreamReader(System.in));
- Random rand = new Random();
- StringTokenizer st = new StringTokenizer(str, delim);

We cannot use the methods from BufferedReader directly we must use r. Likewise for Random and StringTokenizer.
If we were to change the Integer class to this:

```java
public class Integer {
    public int parseInt(String s) {
        int val = 0;
        for (int i = 0; i < s.length(); i++) {
            int lVal = (int) (s.charAt(i) - '0');
            val = val * 10 + lVal;
        }
        return val;
    }
}
```

we would no longer be allowed to use:

```java
Integer.parseInt(str); //This is now BAD!!!
```

We would now have to use:

```java
Integer myInt = new Integer();
myInt.parseInt(str); //This is how we have to do it.
```
public vs private:
When calling methods from other classes we may wish to limit what other classes can see/use...in which case we would use the keyword to private.

We would do this for security reasons, if we only wanted to use a method in a certain way, we could ensure this by setting its visibility/access modifier to private.

We will see when to use this, it is primarily used with instance variables...
Instance variables are variables that are linked to a specific object of that class:

```java
public class Point{
    public int X;
    public int Y;
}
```

```java
public class Test{
    public static void main(String [] args){
        Point P1 = new Point();
        Point P2 = new Point();
        Point P3 = new Point();
        P1.X = 3;
        P1.Y = 4;
        P3.Y = -5;
    }
}
```

<table>
<thead>
<tr>
<th>P1: Point</th>
<th>P2: Point</th>
<th>P3: Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>X = 3</td>
<td>X = 0</td>
<td>X = 0</td>
</tr>
<tr>
<td>Y = 4</td>
<td>Y = 0</td>
<td>Y = -5</td>
</tr>
</tbody>
</table>
We can add in non-static methods to our classes that can act upon our instance variables.

```java
public class Point{
    public int X;
    public int Y;
    public void print(){
        System.out.println(“(”+X+“, ”+Y+“)”);
    }
}

public class Test{
    public static void main(String [] args){
        Point P1 = new Point();
        Point P2 = new Point();
        P1.X = 3;
        P2.Y = -5;
        P1.print();
        P2.print();
    }
}
```
A constructor is a specialized type of method that has exactly the same name as the class and no return-type. It can take any number of parameters. It's job is to initialize the instance variables. It is called right after the new keyword when creating the variable.

```java
public class Point{
    public int X;
    public int Y;
    public Point(){
        X = 10;
        Y = 5;
    }
    public void print(){
        System.out.println("("+X+", "+Y+")");
    }
}
```

```java
public class Test{
    public static void main(String [] args){
        Point p1 = new Point();   //calls the constructor here
        p1.print();               //will print out: (10, 5)
    }
}
```
The class structure generally looks like:

```java
public class <class_name>{
    //instance variables: make them private...use
    //methods to change their values

    //constructor(s)

    //methods
}
```

Let's take a look at several classes.
public class Point{
    private int x;
    private int y;
    public Point(){
        x = 0;
        y = 0;
    }
    public Point(int x1, int y1){
        x = x1;
        y = y1;
    }
    public int getX(){
        return x;
    }
    public int getY(){
        return y;
    }
    public void print(){
        System.out.println("("+x+", "+y+")");
    }
    public Point getMid(Point p2){
        int xm = (x + p2.getX())/2;
        int ym = (y + p2.getY())/2;
        return new Point(xm, ym);
    }
}
public class Rectangle{
    private int width;
    private int height;
    public Rectangle(Point tl, Point br){
        width = br.getX() - tl.getX();
        height = tl.getY() - br.getY();
    }
    public int getWidth(){
        return width;
    }
    public int getHeight(){
        return height;
    }
    public void print(){
        System.out.println("width: "+width);
        System.out.println("height: "+height);
        System.out.println("area: "+getArea());
    }
    public int getArea(){
        int area = width * height;
        return area;
    }
}
public class Circle{
    private Point center;
    private int radius;
    public Circle(){
        center = new Point();
        radius = 0;
    }
    public Circle(Point c, int r){
        center = c;
        radius = r;
    }
    public int getRadius(){
        return radius;
    }
    public Point getCenter(){
        return center;
    }
    public void print(){
        System.out.print("Center: ");
        center.print();
        System.out.println("radius: "+radius);
        System.out.println("area: "+getArea());
    }
    public double getArea(){
        double area = radius * radius * 3.14159;
        return area;
    }
}
public class Shapes{
    public static void main(String [] args){
        Point p1 = new Point();
        Point p2 = new Point(4, 8);
        Point p3 = p1.getMid(p2);
        Point p4 = p3. getMid(p2);
        Rectangle rect = new Rectangle(p3, p4);
        Circle circl1 = new Circle();
        Circle circl2 = new Circle(p3, 10);

        p1.print();
        p3.print();
        rect.print();
        circl1.print();
        circl2.print();
    }
}

//What is printed?
Write an Employee class that has attributes for his name, position, and hourly salary. It should have two constructors: (one that takes no parameters), (one that takes parameters to set the instance variables). It should have a method that prints their information, and one that returns the amount he gets paid based on a number of hours, and methods that will return the instance variable information (get methods)...

public class TestEmployee {
    public static void main(String [] args){
        Employee emp1 = new Employee();
        Employee emp2 = new Employee("John Doe", "Clerk", 6.50);

        System.out.println("Employees:");
        emp1.print();
        emp2.print();

        System.out.println(emp2.getName() + " makes " +
                emp2.getPay(10)+"for 10 hours of work as a "+
                emp2.getTitle());
    }
}