Classes and Objects Continued

Recall from the previous lecture:

```java
Employee employee1 = new Employee(); // new object is created
employee1.setName("Jane Smith");
employee1.setPayRate(9.50);
employee1.resetUnpaidHours();
```

That's rather long. Is there a better way? Yes, we can define a Constructor method for class Employee, so that all of the above would be done in one call:

```java
Employee employee1 = new Employee("Jane Smith", 9.50, 0);
```

Each Object is an instance of its Class.
Has specific values for all attributes
Constructor:

- special kind of a method, has the same name as the class
- are called when an object is created with new
- if a class does not define a constructor – Java automatically creates a default constructor,
  - default constructor takes no parameters,
  - just allocates memory for the object, without initializing any instance variables
  - Example: Employee() is a call to a default constructor in the following
    ```java
    Employee employee1 = new Employee();
    ```
- If any constructor is provided, then no constructors are created automatically

Defining a constructor:

- If any constructor is provided, then no constructors are created automatically
- Constructor methods have the same name as the class
- Have no return type definition in the header
- Actually return a reference to a created object

Example – constructor for the class Employee

```java
/*
 * Constructor. Sets the name attribute of created object to aname,
 * payRate to rate and unpaidHours to hoursWorked.
 */
public Employee (String aname, double rate, double hoursWorked ) {
    name = aname;
    payRate = rate;
    unpaidHours = hoursWorked ;
}
```

If Employee class is augmented with this constructor, the following could be used to create an Employee object:

```java
Employee employee1 = new Employee(“Jane Smith”, 9.50, 0);
```

If now tried to use

```java
Employee employee1 = new Employee();
```

would get an error message from compiler (why?)
Good programming style:

- Include a constructor that initializes all instance variables
- Include a constructor that has no parameters (your own, not JAVA defined default constructor)

/*
 * The default Constructor. Sets the name attribute
 * of created object to Unknown,
 * payRate and unpaidHours to 0.
 */

public Employee () {
    name = "Unknown";
    payRate = 0;
    unpaidHours=0;
}

If this default constructor is added to the Employee class, the following code would create an Employee object with name “Unknown”

    Employee employee1 = new Employee();
Modified definition: Employee.java

```java
public class Employee
{
    // Instance variables of class Employee
    private String name; // name of employee
    private double payRate; // hourly pay rate
    private double unpaidHours; // number of hours for which the pay is due

    // constructors:
    
    /*
    * Constructor. Sets the name attribute of created object to aname,
    * payRate to rate and unpaidHours to hoursWorked.
    *
    */
    public Employee (String aname, double rate, double hoursWorked )
    {
        name = aname;
        payRate = rate;
        unpaidHours = hoursWorked ;
    }

    /*
    * The default Constructor. Sets the name attribute
    * of created object to Unknown, payRate and unpaidHours to 0.
    */
    public Employee ( ) {
        name = "Unknown";
        payRate = 0;
        unpaidHours=0;
    }

    // Instance methods same as in the previous version

    . . . . . .
}
```
Using the class Employee.java in a program EmployeeConstructorsDemo.java (must be compiled together with Employee.java)

This example demonstrates the two ways in which an object of class Employee can be created and initialized.

Also demonstrates a static method WhoEarnedMore that has input parameters and return value of class type Employee.

```java
class EmployeeConstructorsDemo {
    public static void main (String args[]) {
        // use the first constructor
        Employee employee1 = new Employee("Jane Smith", 9.50, 0);
        // new object is created
        System.out.println("How long did "+ employee1.getName() + " work today?");
        double hrs = SavitchIn.readLineInt();
        employee1.IncrementHoursBy(hrs);

        // use the second constructor
        Employee employee2 = new Employee();
        employee2.PrintEmployeeInfo();
        Employee employee3 = employee2;
        employee3.setName("George Clark");
        employee3.setPayRate(12);
        employee3.IncrementHoursBy(10);
        employee3.PrintEmployeeInfo();
        employee2.PrintEmployeeInfo();
        Employee empEarnedMore = WhoEarnedMore(employee1, employee2);
        System.out.println("The person (or one of the people) who earned the largest amount is "+ empEarnedMore.getName());
    }

    /* note: this method illustrates passing objects as parameters and using them as returned values */
    public static Employee WhoEarnedMore(Employee emp1, Employee emp2)
    {
        Employee emp;
        if (emp1.AmtDue() >= emp2.AmtDue())
            emp = emp1;
        else
            emp = emp2;
        return emp;
    }
}
```