Welcome

CSC-285 Advanced Java Programming

Muhammad Javed, Ph.D.

Adjunct Faculty (BHCC)

Webcam ON/OFF

- College doesn't have a policy around it.
- I encourage you to turn your cameras on if you feel comfortable.
- At least when asking questions or making comments.

- So, we can know each other better.

Week 1

RECAP - I

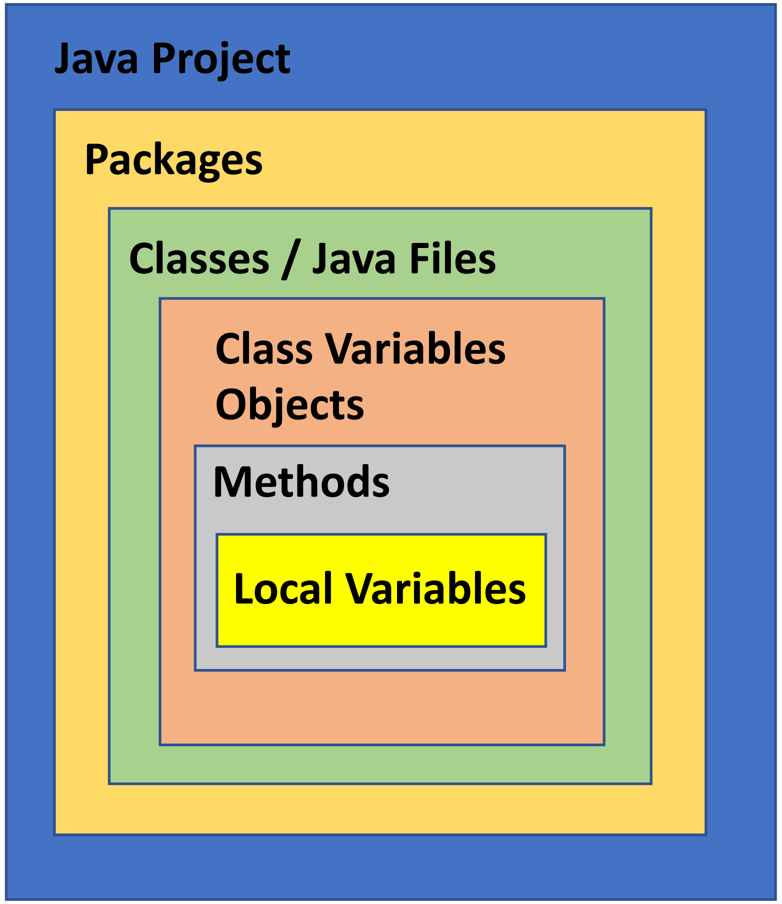
- **Elements of a Java Program**
- **Classes and Objects & Methods**
- **Variables and Data Types**
- **Running first Program (Eclipse)**

Elements of a Java Program

Classes, Objects & Methods

Classes, Objects & Methods

- A Java program is composed of classes
- A class consists of objects, methods and variables
- Objects often represent real-world entities



Classes, Objects & Methods

- **For example:**
  - **A *Car* class, a *Robot* class**
  - **All cars can be represented by a class called *Car***
  - **All robots can be represented by a class called *Robot***
- **Specific cars (your car, my car, John's car) can be represented by objects (a.k.a. instances) of this *Car* class.**
- **Specific robots can be represented by objects of the *Robot* Class (e.g. Robot1, Robot2, ...)**

Robot Class

```
public class Robot {

   private String myName = "nobody";

   public void setName (String name) {
      myName = name;
   }

   public void sayHelloTo (String name) {
       System.out.println("Hello " + name + "!");
       System.out.println("I'm Robo" + myName + ".");
   }
}
```

- Class Name + declaration
- Class body
- Variable name
- Variable value
- Method names

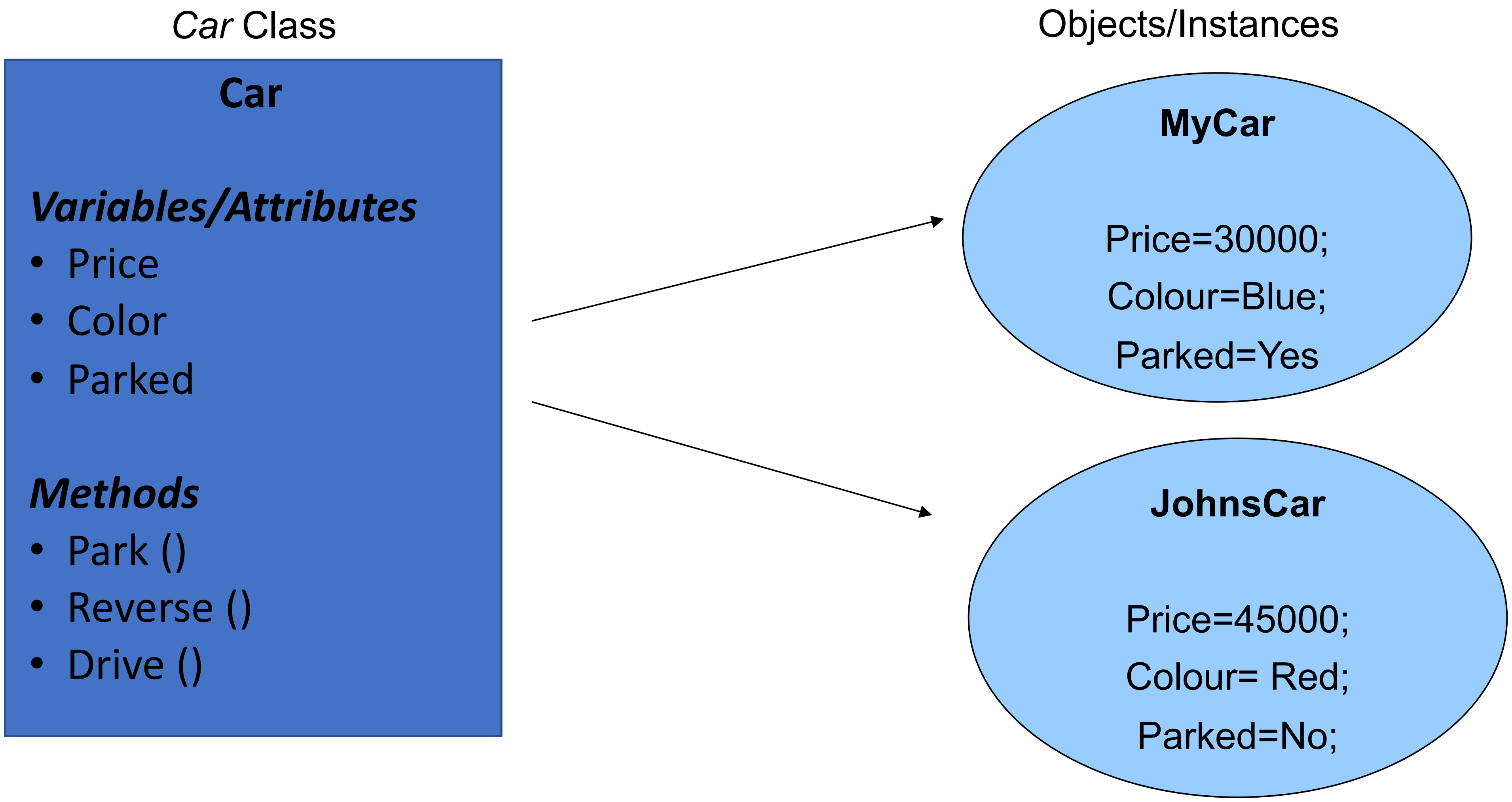
Classes, Objects & Methods

- Objects often represent real-world entities
- A class is a template that a developer writes.
- An object is created (or *instantiated*) from the class.
  - This process is called instantiation.
- The objects of a class are also called the instances of that class.

Classes, Objects & Methods

- **Each object has a *state***
- **State can be considered as a set of characteristics e.g.**
  - **my car is *blue,* its price is *$30,000*, and it is currently *parked***
  - **John's car is *red,* its price is *$45,000,* and it is currently *not parked,* etc.**

Classes, Objects & Methods



Car Class

Car

Variables/Attributes

- Price
- Color
- Parked

Methods

- Park ()
- Reverse ()
- Drive ()

Objects/Instances

MyCar

Price=30000;

Colour=Blue;

Parked=Yes

JohnsCar

Price=45000;

Colour= Red;

Parked=No;

Classes, Objects & Methods

- **Each object also has state and *behavior***

- **For example, the car can be parked, can be driven, etc.**

- ***State* is represented by the values of its data items, represented by variables**

- ***Behavior* is represented by what are called methods**

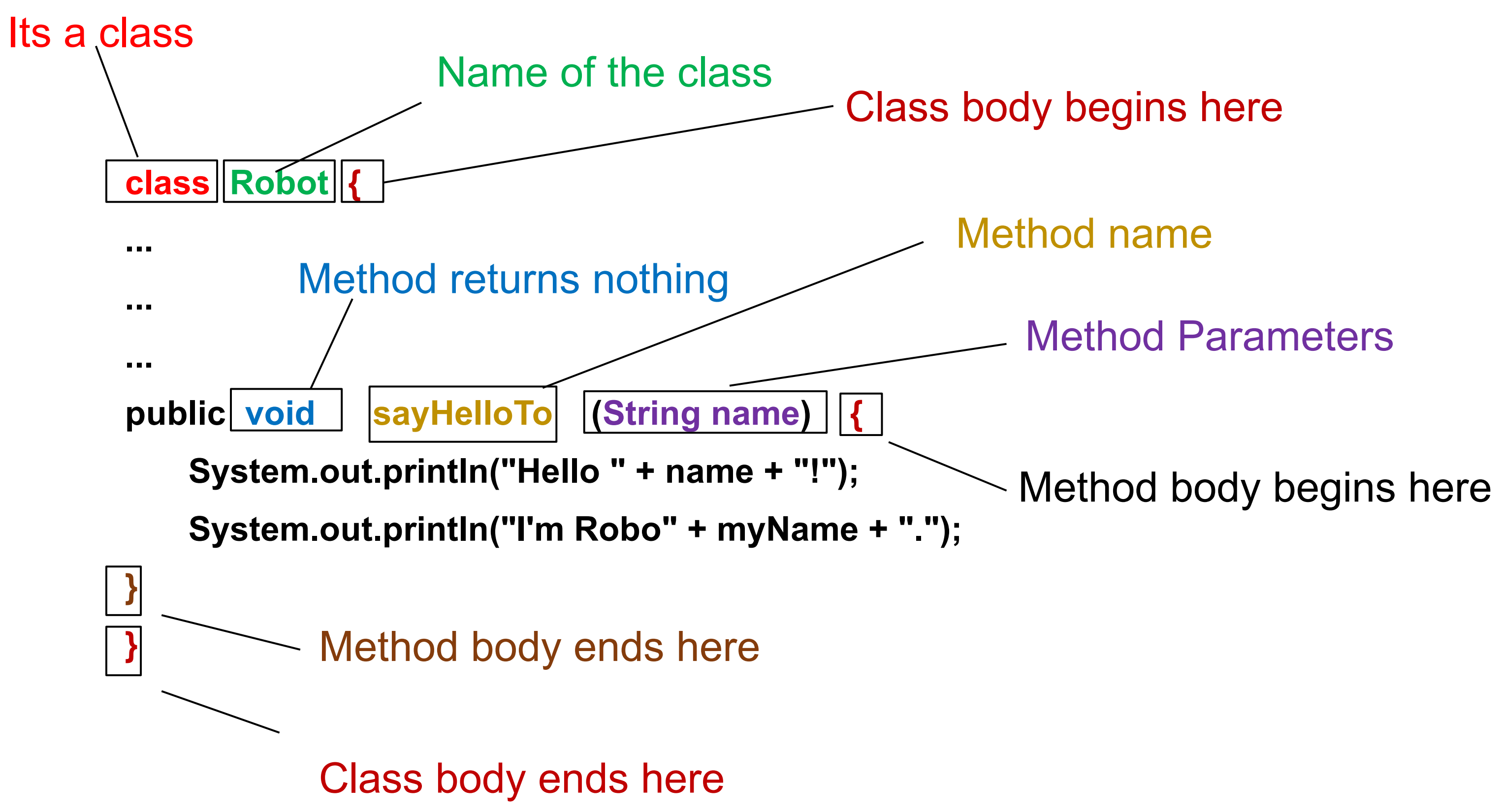
Classes, Objects & Methods

- **A program, written in any programming language, is basically made up of three elements:**
  - **Data**
  - **Operations on data**
  - **Logic that determines the operations**

- **Operations on data and Logic for the operation are held inside a method that determines the behavior of an object.**

- **Like classes, methods have a *declaration* and a *body***

Classes, Objects & Methods



Its a class

Name of the class

Class body begins here

```
class Robot {
...
...
...
public void sayHelloTo (String name) {
     System.out.println("Hello " + name + "!");
     System.out.println("I'm Robo" + myName + ".");
}
}
```

Method returns nothing

Method name

Method Parameters

Method body begins here

Method body ends here

Class body ends here

Classes, Objects & Methods

- **A method is executed by specifying the method name (and the name of the object to which the method belongs) e.g.**

- **Robot robot = new Robot(); // creates an object**
- **robot.sayHelloTo(“John”); // invokes a method**

- **Executing a method is also called *calling* or *invoking* the method.**

Accessing Classes & Methods

public

- a class member declared *public* can be accessed by any code from any class in your application
- An application declares its main(…) method to be public so that it can be invoked from any JVM
- This modifier makes a class or a class member *most accessible*.

protected

- A member declared *protected* is accessible to all classes in the same package in which the class that declares the member exists.
- The protected member can also be accessed from a subclass of the class that contains the member, even if the subclass is in a different package.
- The protected modifier provides *less accessibility than the public modifier*.

default

- *If you do not specify any access modifier* while declaring a class or a class member, the default access is assumed.
- Can be accessed by any class in the same package as the class in question.
- Provides *less accessibility than the protected* modifier.

private

- A member declared private is *only accessible to objects of the same class* in which the member is declared.
- A top-level class cannot be declared private.
- This modifier makes a class member *least accessible*.

Elements of a Java Program

Variables and Data Types

Variables and Data Types

- A variable can be considered a symbol that represents (or in other words points to) a value stored in the computer memory

- The value represented by a variable can be changed (hence the name “variable”)

- A variable has a declaration. You must specify the variable type and name

- The declaration of a variable, i.e. declaring its name and type, looks like the following:

```
        <type> <name>;
```

- You can also assign an initial value to a variable while declaring it by using the following syntax:

```
        <type> <name> = <value>;
```

Variables and Data Types

- The name must be a legal identifier
  - Not one of the ***reserved names*** (or ***keywords***) of the Java language

- The following rules determine a legal name:
  - The name in general begins with a letter (a–z, A–Z) or _
  - Can not begin with a digit
  - The first character of the name can be followed by a series of letters, _, $, or digits, where a digit is 0–9 or any Unicode character that denotes a digit in a language

Class vs. Local Variables

Class vs. Local Variables

Encapsulation and Data Hiding

- **A class in Java can be looked upon as a basic unit of encapsulation.**

- **A variable declared outside of a method is called a class variable or an instance variable.**

    - **A variable declared inside a method is called a local variable**

- **These instance/class variables and the methods of a class are called *members* of the class.**

Variables and Data Types

For example:

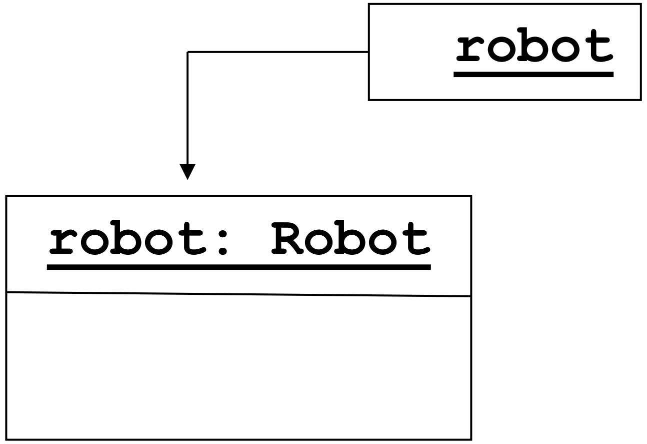
```
      int luckyNumber = 7;
```

- The equal sign (=) assigns the number 7 to the variable `luckyNumber`
- The equal sign is an example of what are called assignment operators.
- The int type is one of several basic data types called primitive data types

Variables and Data Types

- Classes in Java are also considered data types, and you can declare variables of this type as well.
- For example, following rows declare and instantiate an instance of Robot Class:

```
Robot robot;
robot = new Robot();
```



- **Robot** is the name of the class and **robot** is the name of a variable of type Robot
- Such a variable is called an Object reference variable, because it is created to refer to an object

```
Robot robot;
robot = new Robot();

robot = new Robot(“Bumble Bee”);

robot = new Robot(“Bumble Bee”, “45”);
```

Q: What these three lines mean ?

Primitive Data Types in Java

- `boolean`: This data type is used to represent a binary condition: *true* or *false*.

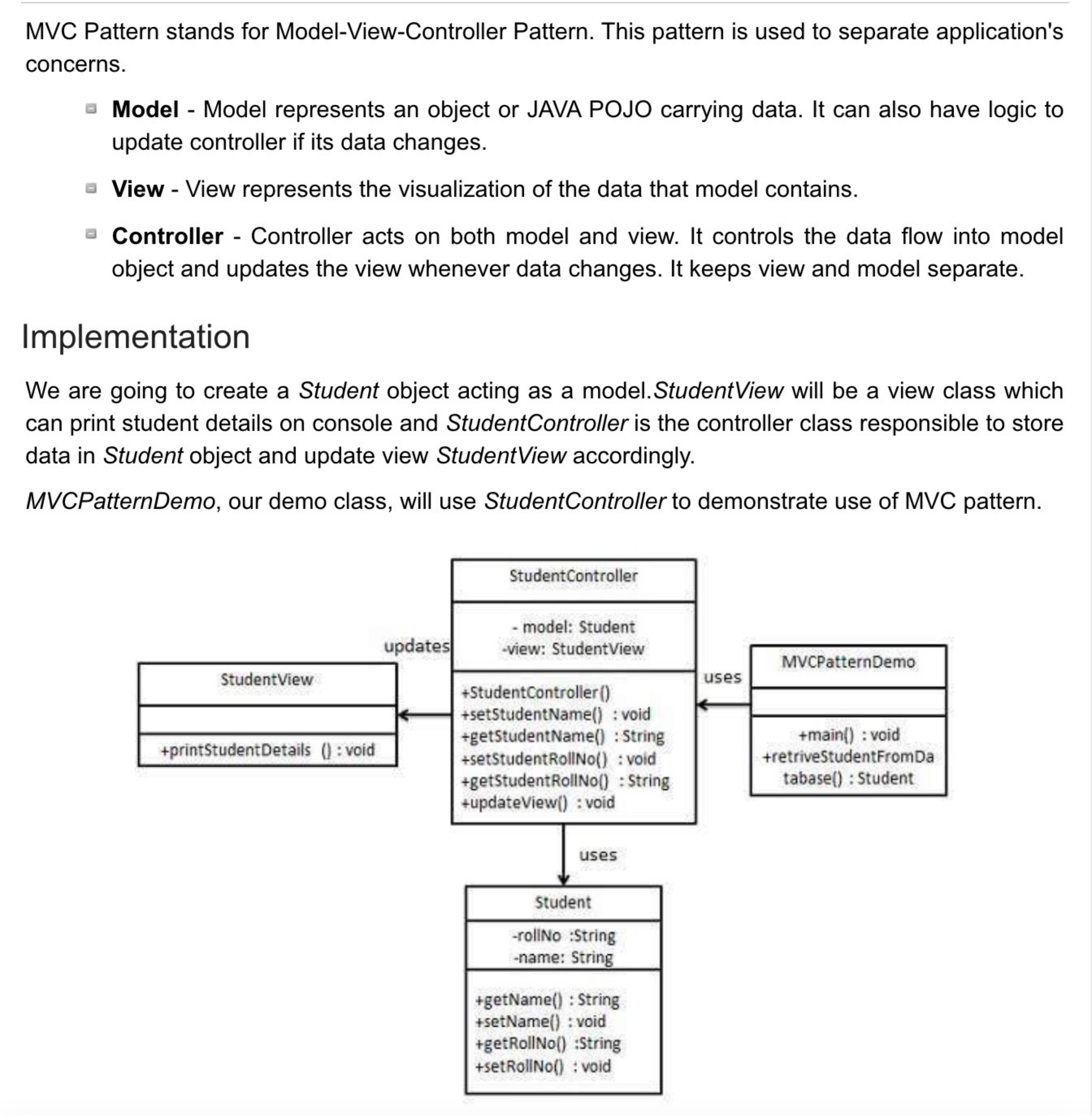
- `char`: This type is a 16-bit, *unsigned* integer that is used to represent keyboard characters.

- `byte`: This type is an 8-bit, signed, two’s complement integer.
- `short`: This type is a 16-bit, signed, two’s complement integer.
- `int`: This type is a 32-bit, signed, two’s complement integer.
- `long`: This type is a 64-bit, signed, two’s complement integer.

- `float`: This type can hold a 32-bit, signed floating-point number.
- `double`: This type can hold a 64-bit, signed floating-point number.

Style Guide

- *JavaDoc-specific Comments*
- *Modifiers (public, private, protected)*
- *Indentation*
- *Meaningful Variable|Class|Method names.*
- *MVC - Design Patterns*



MVC Pattern stands for Model-View-Controller Pattern. This pattern is used to separate application's concerns.

- **Model** - Model represents an object or JAVA POJO carrying data. It can also have logic to update controller if its data changes.
- **View** - View represents the visualization of the data that model contains.
- **Controller** - Controller acts on both model and view. It controls the data flow into model object and updates the view whenever data changes. It keeps view and model separate.

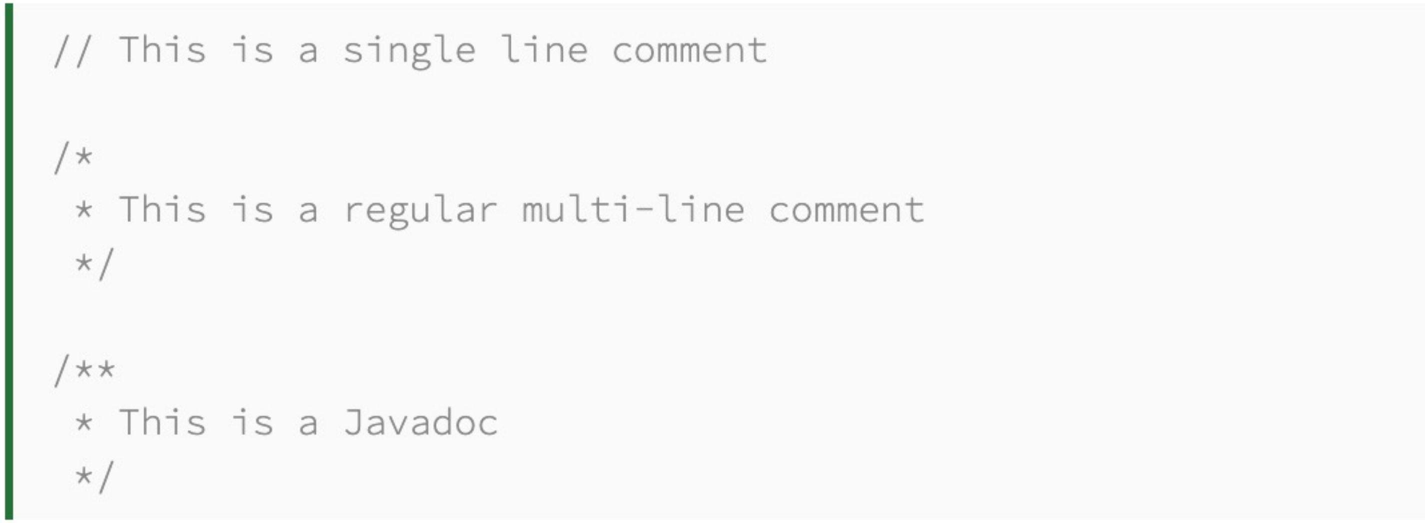
Implementation

We are going to create a *Student* object acting as a model.*StudentView* will be a view class which can print student details on console and *StudentController* is the controller class responsible to store data in *Student* object and update view *StudentView* accordingly.

MVCPatternDemo, our demo class, will use *StudentController* to demonstrate use of MVC pattern.

Taken from: https://www.tutorialspoint.com/design_pattern/mvc_pattern.htm

Documentation in Java



```
// This is a single line comment

/*
 * This is a regular multi-line comment
 */

/**
 * This is a Javadoc
 */
```

Content taken from https://www.baeldung.com/javadoc

JavaDoc at Method Level

```
/**
* Write the description of the method here
* Spread the description
* To multiple lines
* @param <parameter name> Description of the parameter 1
* @param <parameter name> Description of the parameter 2
* @return the description of the the returned value/object
*/
public int getArea(int height, int length) {
        // do things

        return area;
}
```

JavaDoc at Method Level

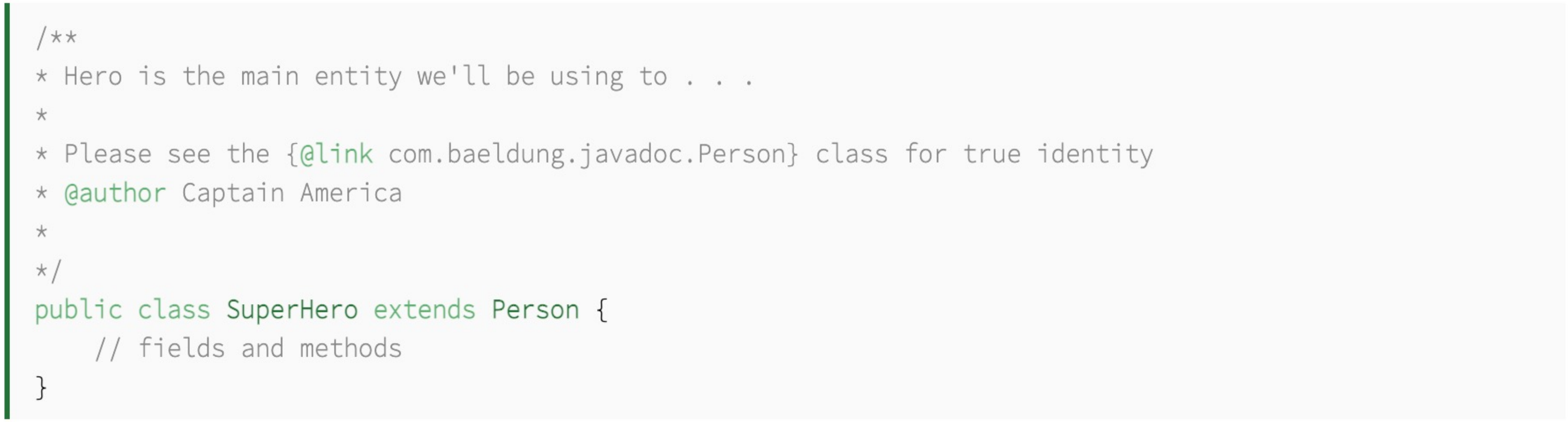
```
/**
* This method calculates the area of a rectangle.
* The height and the length values are passed as
* input parameters.
* @param height stores the height value
* @param length stores the length value
* @return the method calculates and return the area of the rectangle.
*/
public int getArea(int height, int length) {
        // do things

        return area;
}
```

@link

@param

@return

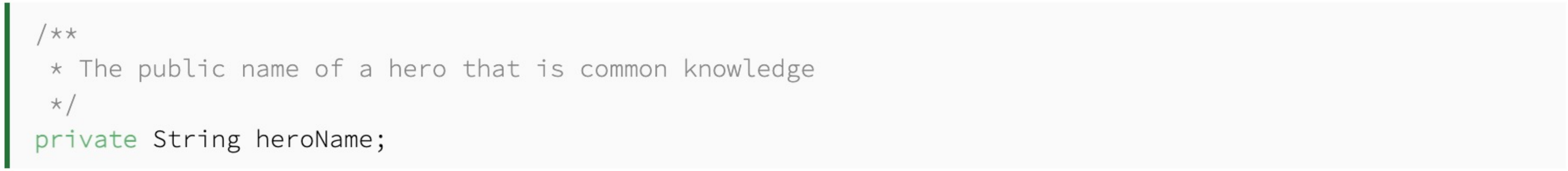
JavaDoc at Class Level



```
/**
* Hero is the main entity we'll be using to . . .
*
* Please see the {@link com.baeldung.javadoc.Person} class for true identity
* @author Captain America
*
*/
public class SuperHero extends Person {
    // fields and methods
}
```

- **@link**
- **@author**

JavaDoc at Class Variable Level



```
/**
 * The public name of a hero that is common knowledge
 */
private String heroName;
```

Java Program I

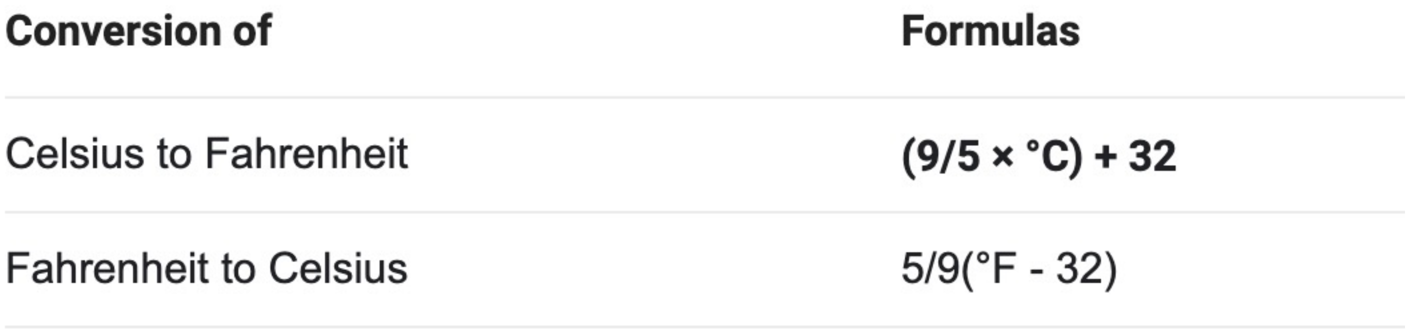
- **Write a Java Method "*equationSolver()*" to calculate (x2 – x1)^2 + (y2 – y1)^2**

- It set x2 = 4, x1 = 2, y2 = 6 and y1 = 3.
- Pass x1, x2, y1, y2 values as an *parameters* to the Method.
- Print the calculated results.

Java Program II

- **Fahrenheit to Celsius / Celsius to Fahrenheit**

- Write a program with two methods.
  - Method that converts Fahrenheit to Celsius
  - Method that convert Celsius to Fahrenheit



Conversion of	Formulas
Celsius to Fahrenheit	**(9/5 × °C) + 32**
Fahrenheit to Celsius	5/9(°F - 32)