Object Oriented Programming in JAVA

Object Oriented Programming

Objective: Introduction to Object Oriented Programming, to become familiar with Objects and classes, declaring a class with instance variables and methods and Instantiating an object of a class, and become familiar with:

- **instance variables and methods**
- **access modifiers : private, public and protected**
- **Accessors and Mutators getters() and setters() .**
- **Constructor and Destructor.**
- **Static (class) variables**

Object Oriented Programming

- **What is a Programming Paradigm ?**

Definition: A **programming paradigm** is a style or "**way**" of programming.

Some of the more common paradigms are:

- **Structured** — Programs have clean, goto-free, nested control structures.
- **Array-based** —
- **Functional** Programming

Object Oriented Programming

- Some of the more common paradigms are:
- **Event-Driven** – Control flow is determined by asynchronous actions (from humans or sensors).
- **GUI-Based**

Programming Paradigm

- **Imperative** — Control flow is an explicit sequence of commands.
- **Procedural** — Imperative programming with procedure calls.
- **Flow-Driven** — Computation is specified by multiple processes communicating over predefined channels.
- **Logic (Rule-based)** — Programmer specifies a set of facts and rules, and an engine infers the answers to questions.

Programming Paradigm

- **Constraint** — Programmer specifies a set of constraints, and an engine infers the answers to questions.
- **Aspect-Oriented** — Programs have cross-cutting concerns applied transparently
- **Object-Oriented** — Computation is effected by sending messages to objects; objects have state and behavior.
  - **Class-based** — Objects get their state and behavior based on membership in a class.

Introduction to Object-Oriented Programming

- **Object-Oriented Programming :** is a Programming paradigm **or** a software development methodology in which a program is conceptualized as a group of objects .
- A Programming technique in which a program is organized as a collection of objects.
- A software development approach for writing and developing computer programs based on objects.

Introduction to Object-Oriented Programming

- OOP provides an easy of modeling things in the real world.
- OOP organizes & combines data and functions together, hides and encapsulates the implementation details and offers several features for developing computer softwares.

- Object-oriented programming is the most dramatic innovation in software development.
- Integrating data and functions (methods) is the central idea of object-oriented programming.
- Using object-oriented programming we can write clearer, more reliable, more easily maintained programs.

- The class is the foundation of Java’s support for object-oriented programming.
- The class provides the mechanism by which objects are created. Thus a class defines a new data type, which can be used to create objects.
- A class is created by using the keyword class.

- **Object-oriented programming aims to achieve the following:**
- To simplify the design and implementation of complex programs.
- To make it easier for teams of designers and programmers to work on a single software project.
- To enable a high degree of reusability of designs and of software codes.
- To decrease the cost of software maintenance.

Introduction to Object-Oriented Programming

- **OOP : makes the development process easier faster and less time consuming.**
- **OOP features :**
  - Classes and Objects
  - Data Abstraction & Encapsulation
  - Inheritance and Code-reusability
  - Polymorphism
  - Interfaces
- And a programming language that supports or follows these features is called an Object-Oriented Programming Language.
- Java is a an OOP language.

Introduction to Object-Oriented Programming

- **Apart from these concepts, there are some other terms which are used in Object-Oriented programming:**
- **Coupling**
- **Cohesion**
- **Association**
- **Aggregation**
- **Composition**

And a programming language that supports or follows these features is called an Object-Oriented Programming Language.

Java is a an OOP language.

- **Program Modules in in java**

Three kinds of modules exits in java- **methods, classes and packages**.

Java Programs are written with predefined **methods and classes available in the Java Application Programming Interfaces** (also referred to as the Java API or Java class library).

Related classes are grouped into packages so that they can be imported into programs and reused.

The Java API classes are part of the JDK.

OOP Classes and Objects

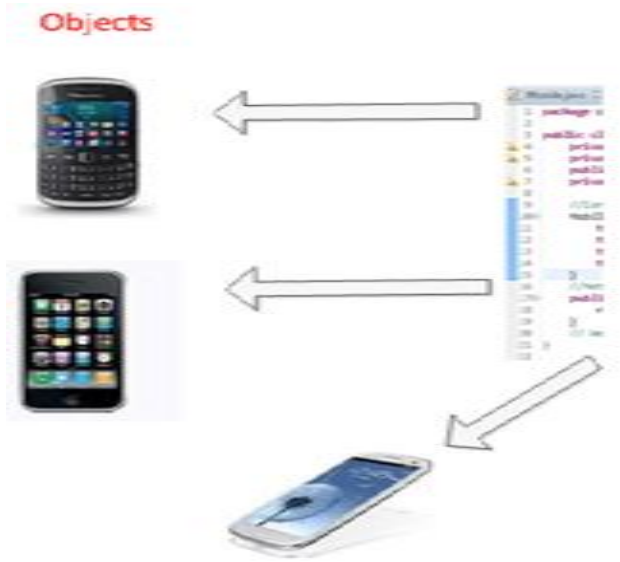
- Two key concepts in OOP are :
- **objects and classes**
- **Object** An **object** is and entity that has certain attributes or properties sometimes called characteristics which may be assigned values The values can be numeric or non-numeric.

  In OOP an Object is said to be an **instance** of a class or working copy of a class or

  Look like a class.
- Objects in a program can represent real world things or abstractions e.g. automobiles, Mobiles, Books houses and employees.
- An Object can be defined as an instance of a class. An object contains an address and takes up some space in memory.

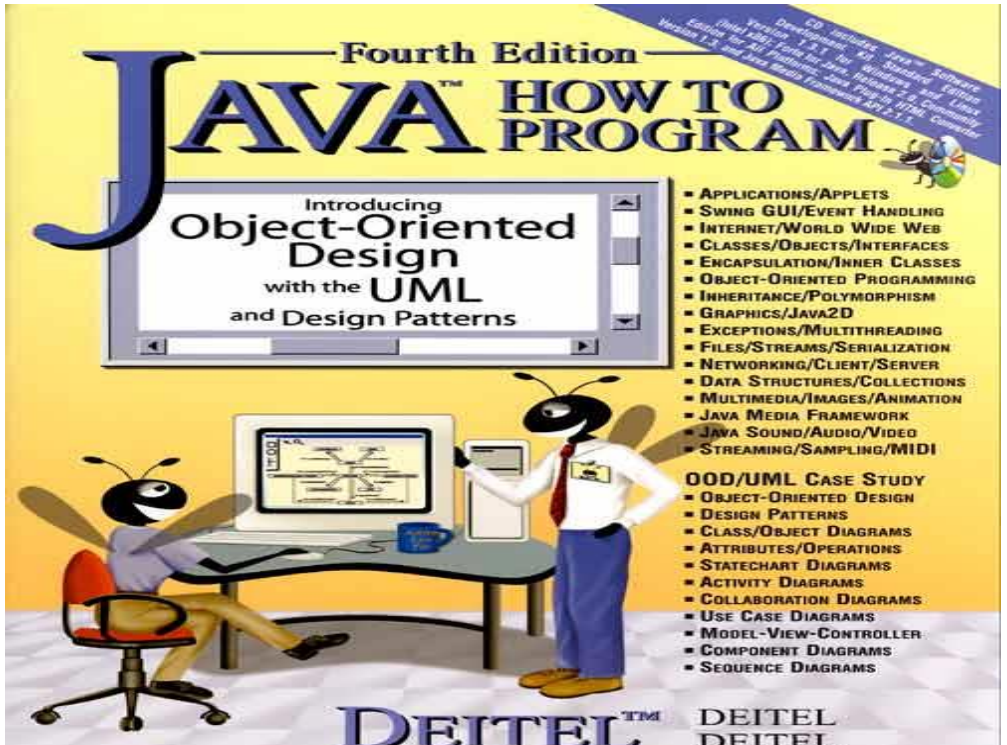
Introduction to Object-Oriented Programming

- **An Analogy**
- **Examples of real world Objects**
- **In the real world everywhere we look we see objects such :**



Introduction to Object-Oriented Programming

- An object represents a real world entity
- **Following figure shows some real-world entities**



Book object

Students

Object

Introduction to Object-Oriented Programming

Book object

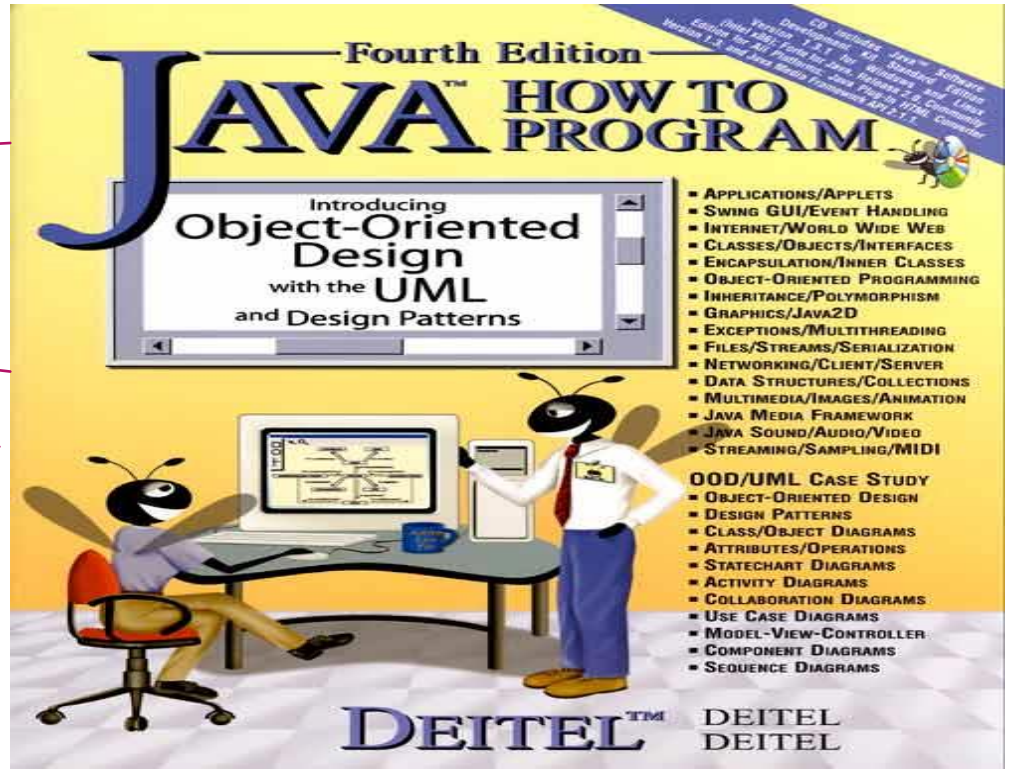
Attributes/Properties

BookName

AuthorName

Price

Edition



Introduction to Object-Oriented Programming

car object

Attributes

Name

Model

Price

Color

Introduction to Object-Oriented Programming

Students

Attributes

Object

Name

RollNo

Address

Introduction to Object-Oriented Programming

Attributes with values

car object

Name=Alto

Model=2017

Price=

Color=white

Introduction to Object-Oriented Programming

Attributes with values

Students

Object

Name=Ali

RollNo=17Sw

Address=xyz

Introduction to Object-Oriented Programming

Attributes

Name

Model

Price

Color

Behavior

Change gear()

Apply breaks()

Accelerate()

car object

Behavior is equivalent to a method in a program

Introduction to Object-Oriented Programming

Attributes

Name

RollNo

Address

Behavior

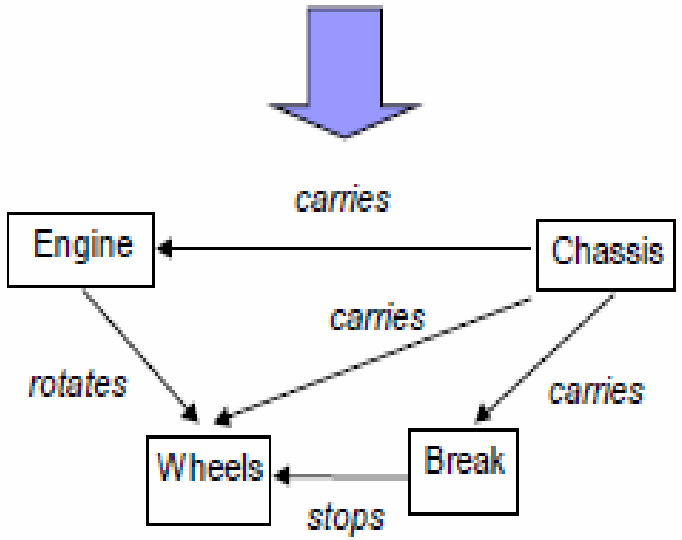
setName()

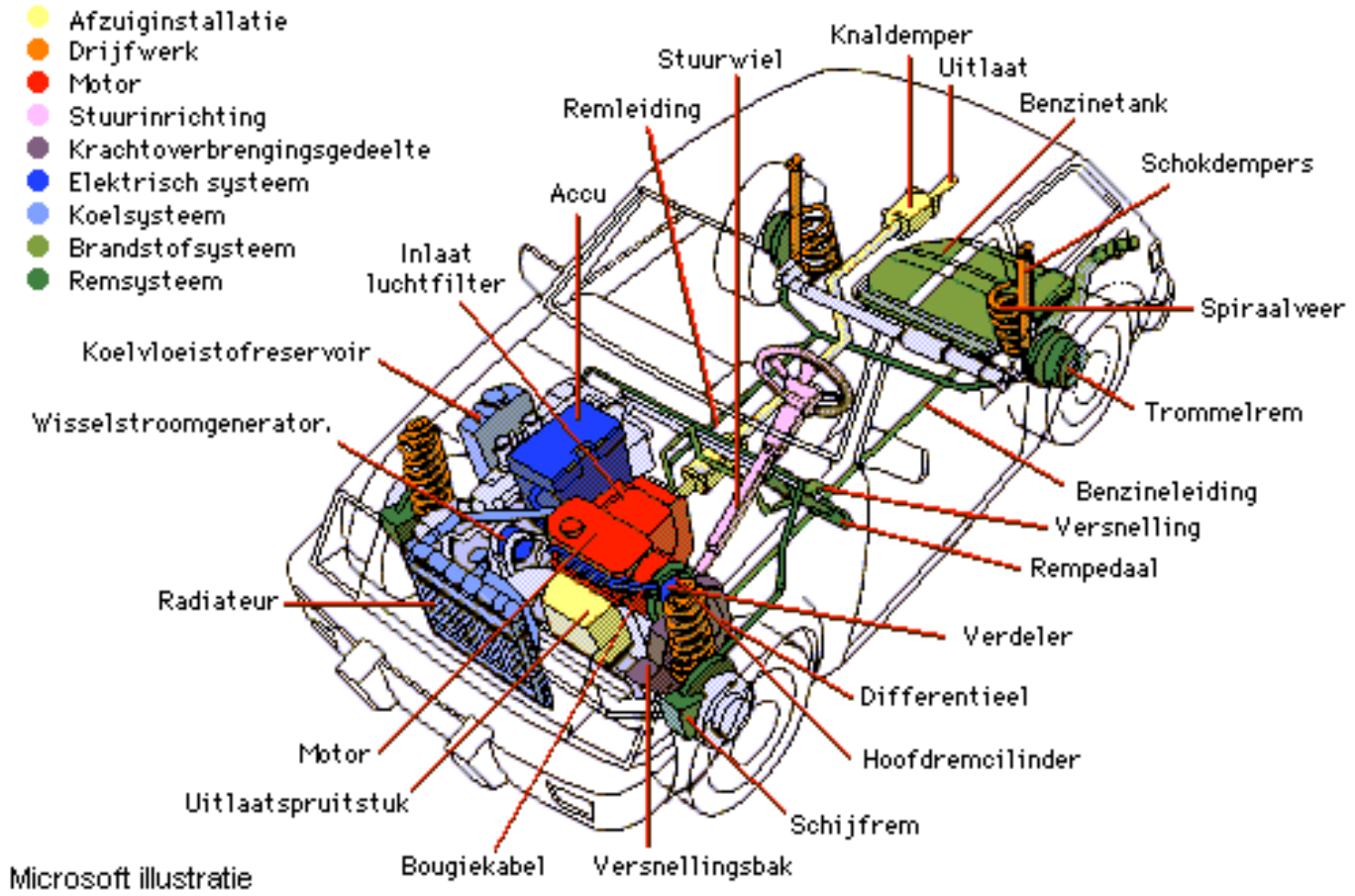
changedept();

setdept();

An Analogy
Classes : A class is a template for an **object**. Or
Classes are the definitions (or blueprints) Used to create objects.
U you can use definitions of class to create objects.

The structure or Design of a car





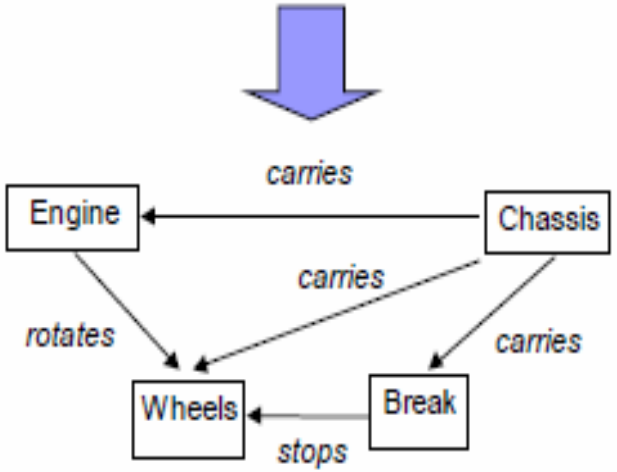
An Analogy

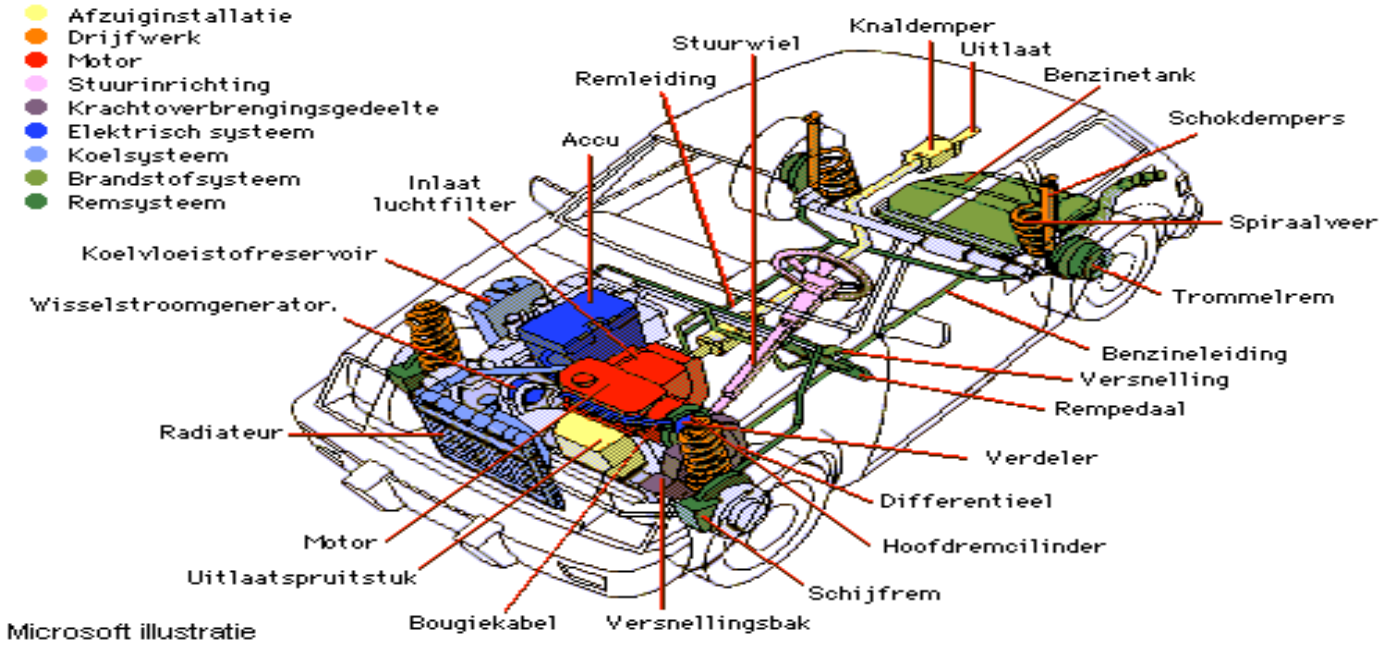
Collection of objects is called class. It is a logical entity.

A class can also be defined as a blueprint from which you can create an individual object.

Class doesn't consume any space.

An Analogy





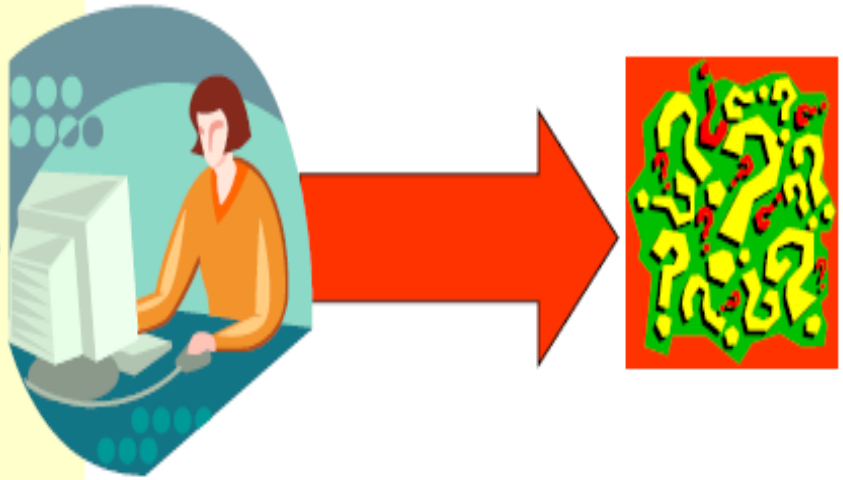
Object 1

Object 2

Coding only will not do...

Coding only will not do...

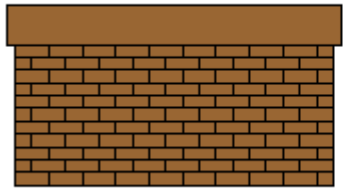
- Lists
- Arrays
- Class
- Object
- Procedures
- Functions
- Algorithms
- Etc.



More programmers...?

Build a Palace…

- Need a Palace…
- Get some bricks,wood…
- Go find many masons, carpenters as you can.
- Build the palace….?



You need an architect first...

Architecture Design

Architecture

Architecture Realization

Building

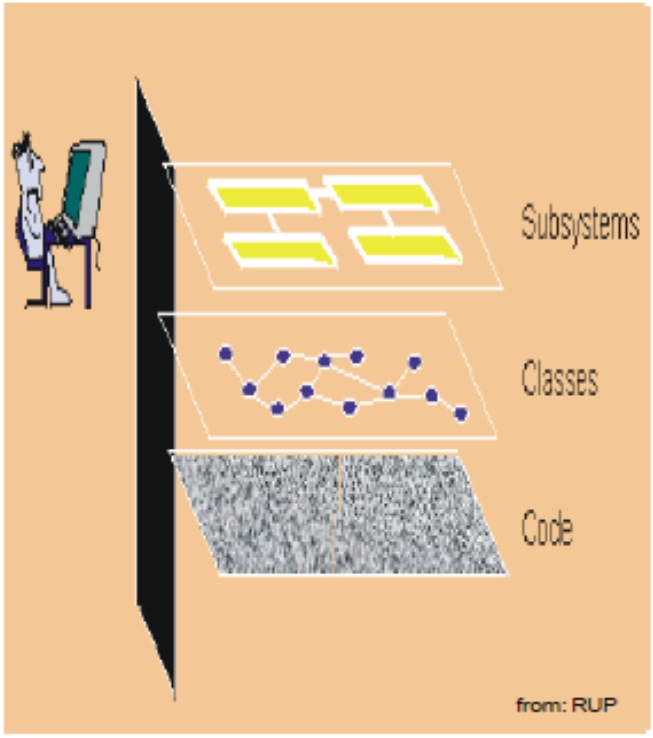
Taj Mahal

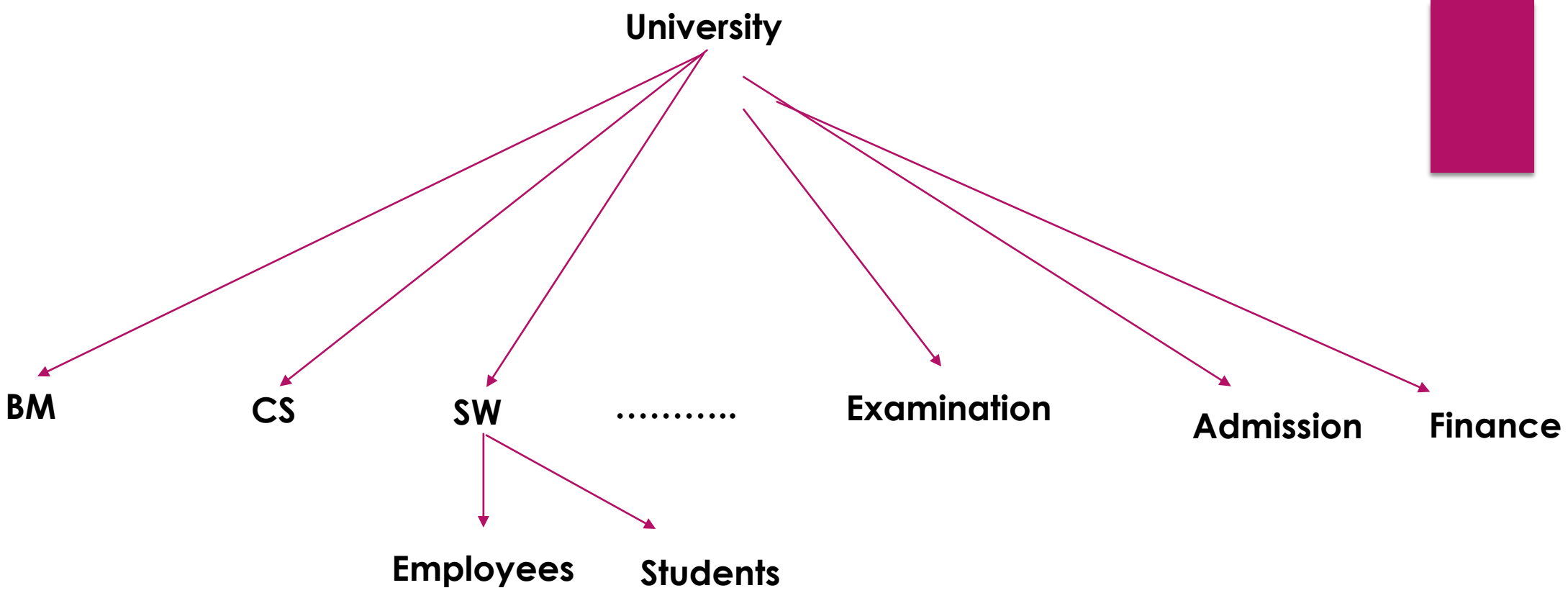
Taj Mahal : Built by Mogol Shah Cihan in memory of his beloved wife Mümtaz Mahal 1648

Coding is flat only in **trivial** systems.

Abstract Specification

- Architecture represents a **high level abstract specification**.
- Abstraction helps to cope with **complexity**
- Abstraction **improves understanding** of the software system.




University
BM
CS
SW
.....
Examination
Admission
Finance
Employees
Students

OOP Class and Object

Class: A class is a template for an **object**. **OR**

- **Classes** are the definitions (or blueprints) Used to created objects.
- **Object :** in general an **object** is an entity that has certain attributes or properties which may be assigned values The values can be numeric or non-numeric.
- In **OOP** an O**bject** is an **instance** of a class **OR**

  an Object is a working copy of a class **OR** Look like a class.
- Objects in a program can represent real world things e.g. automobiles, Mobiles, Books houses and employees.

OOP Class and Object

- There are just two kinds of things that you can include in a class definition:
- **Fields:** These are variables that store data items .They are also referred to as data members of a class.
- The data, or variables, defined within a class are called **instance variables.**
- **Methods:** These define the operations you can perform for the class —so they determine what you can do to, or with, objects of the class. Methods typically operate on the fields —the data members of the class.
- Collectively, the methods and variables defined within a class are called **members of the class.**

OOP Class and Object

- **Syntax**
- class classname {
- type instance-variable1;
- type instance-variable2;
- type instance-variableN;
- type methodname1(parameter-list) {
- // body of method
- }
- // ...
- type methodnameN(parameter-list) {
- // body of method
- }
- }

```
class student {

String roll_no;
}
```

```
class stduentTest {

public static viod main(String args[]){

    student stdobj=new student();


    stdobj.roll_no =“17sw02”;
} }
```

What is Java Application

Recall that main is a special method that is always called automatically by the JVM when you execute an application.

A class that contains **method main is a java application.**

class student **is not an application** because it does not contain **main.**

Error

Exception in thread main java.lang.NoSuchMethodError: main

Declaring a class with a method and Instantiating an object of a class.

A Simple Class

```
class Students  {

  public void DispayMsg()
{
    System.out.println("Welcome to Class SW-Section-I");
}  // end method

}  // end class
```

Use Students Class

```
class StudentsTest  {


 public  static void main (String args []) {

// Creating an object  and asdign it to objStd

Students objStd=new Students();

{
// calling objStd's method DisplayMsg()

objStd.DisplayMsg();

}  // end main method

}  // end class
```

Declaring a method with a parameter
A Simple Class

```
class Students  {

  public void dispalyName(String stdName)
{
    System.out.println(stdName+“Welcome to Class SW-Section-I”);
}  // end method

}  // end class
```

Use Students Class

```
class StudentsTest  {


 public  static void main (String args []) {

// Creating an object  and asdign it to objStd

Students objStd=new Students();

{
// calling objStd’s method DisplayMsg()

objStd.displatname(“Ali");

}  // end main method

}  // end class
```

OOP Mutators & Accessors

- **Mutators (Set methods) &**
- **Accessors (get methods)**

- These public methods are called set and get methods.
- Allow the client of the class to set(**i.e. assign values**) to the private instance variables , and
- get (**i.e. obtain values**) of private instance variables.

Initializing Objects with Constructors

- **A Constructor :** is a special member function which is called/ executed automatically whenever an object of a class is created.
- When an object of class is created, the constructor of that class is called.
- It is used to Initialize an object, and to initialize the instance variables.
- Automatic Initialization is carried out by a special member function called **Constructor**

Initializing Objects with Constructors

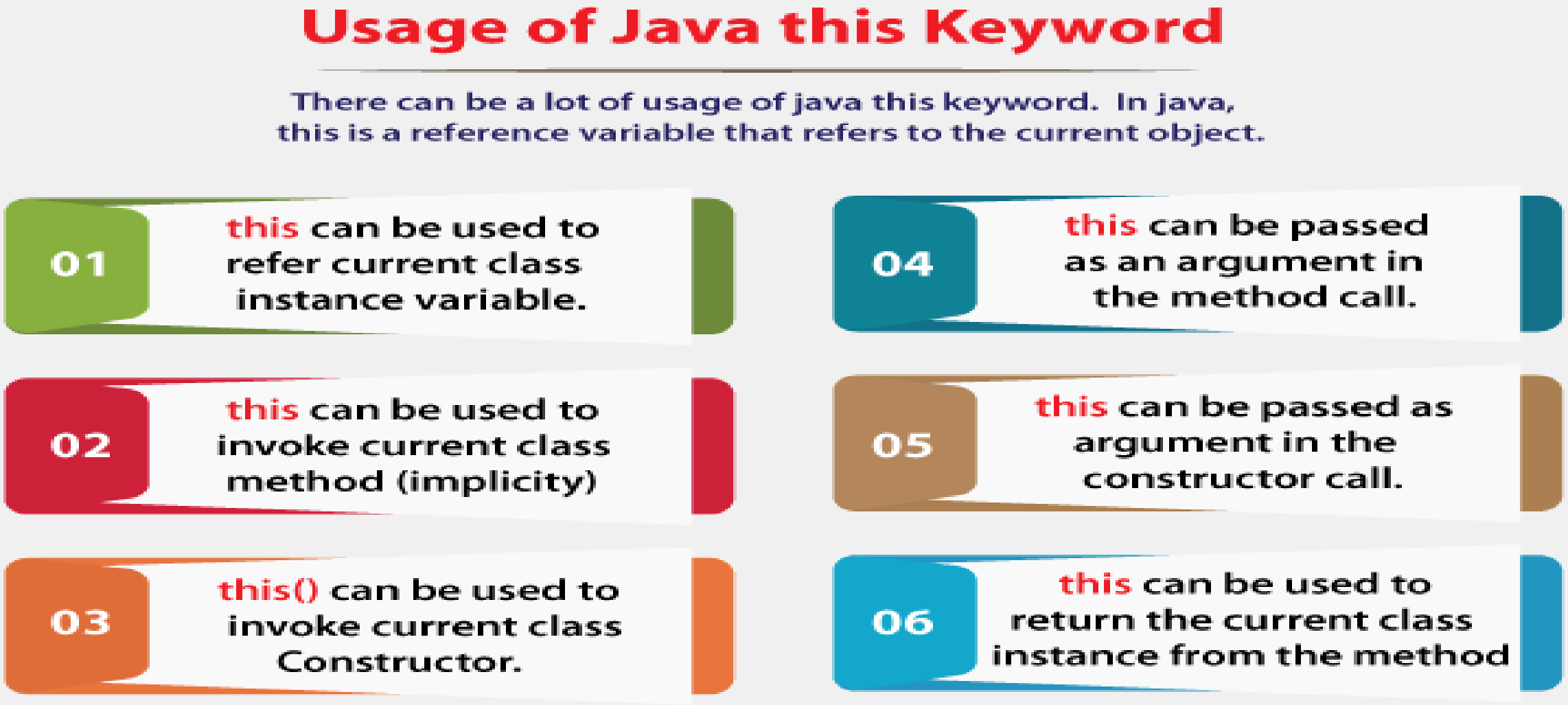
- All classes have a **default constructor which is parameter less**.
- A constructor has **two special characteristics** that differentiate it from other class methods:
  - A constructor always has the same name as the class
  - A constructor never returns a value, and you must not specify a return type —not even of type void.
- If you don't define any constructors for your class, the compiler supplies a default constructor in the class, that initializes the instance variables to their default values (zero for numeric types, false for boolean, null for reference) .

this Keyword

- **this refers** to the current object‘s members**.** You can refers current object with **this** reference**.**
- In Java, this is a reference variable that refers to the current object.
- The compiler uses this implicitly when your method refers to an instance v**ariable of the class.**
- Usage of java this keyword . **6 usage of java this keyword.**

- this can be used to refer current class instance variable.
- this can be used to invoke current class method (implicitly)
- this() can be used to invoke current class constructor.

this Keyword



this Keyword

- **this refers** to the current object‘s members**.** You can refers current object with **this** reference**.**
- The compiler uses **this implicitly when your method refers to an instance variable of the class.**
- Usage of java this keyword . **6 usage of java this keyword.**

- this can be passed as an argument in the method call.
- this can be used to return the current class instance from the method.
- this can be passed as argument in the constructor call.

Controlling access to Members (Access Modifiers)

- **Access Specifiers :** Define level of Accessibility of a variable or a method **OR**
- **The access modifiers** control access to a class's variables and methods.
- Specify which part of the program can access a variable or a method.
- **Private :** variables or methods declared with **access specifier private** are accessible only within a class in which they are declared.
- **Public :** variables or methods declared with **access specifier public** are accessible anywhere in the program (i.e. within a class in which they are declared and also outside that class)
- **Protected** : variables or methods declared with **access specifier protected** are accessible within a class in which they are declared and also in the derived class).

A Simple Class with Constructor

```
class Students  {

String studentdame; // instance variable
//Constructor
// public Students () {
//studentdName=“anyName”) ;}
  public void setName(String stdName)
{
       studentName=stdName;
}  // end method

 public String getName()
{
 return studentName;
}  // end method


 public void displayName()
{
System.out.println(getName());
}  // end method
```

Use Students Class

```
class StudentsTest  {


 public  static void main (String args []) {

// Creating an object  and assign it to objStd

Students objStd=new Students();

{
// calling objStd’s method displayName()
System.out.println(“Initial Value:”+ objStd.getName());

objStd.setName(“Ali");
 System.out.println(“Welcome”+ objStd.getName());

}  // end main method

}  // end class
```

Introduction to Object-Oriented Programming

- **Data Encapsulation & Abstraction**
- Data Encapsulation: Is an OOP feature that binds together data and functions. **OR**
- The wrapping of data and functions into a single unit. **OR**
- The internal of the object is private to that object and may not be accessed from outside the object

- **Abstraction:** refers to the act of representing the functionality of a program and hide the internal details

Introduction to Object-Oriented Programming

- **Data Encapsulation & Abstraction**
- **Data Encapsulation or data hiding**: Is an OOP feature that binds together data and functions. **OR**
- The wrapping of data and functions into a single unit. **OR**
- The internal of the object is private to that object and may not be accessed from outside the object

- **Abstraction:** refers to the act of representing the functionality of a program and hide the internal details. **OR**
- **A feature of OOP where by the implementation details of class are kept hidden from the user.**

Information Hiding

Information hiding is achieved by restricting the access of the user to the underlying data structures to the predefined methods for that class. This shifts the responsibility of handling the data fields correctly from the user of the code to the supplier. For example, a programmer using a Date class is usually not interested in the implementation details. These include the underlying data structure of Date (i.e. whether the month is stored as an integer, a string or an enumeration type) and the underlying code (i.e. how two dates are subtracted from each other). If, at a later stage a more efficient way of storing the date or of calculating the number of days between two dates is introduced, this should not affect the programmers who have been using the date class.

A further advantage of information hiding is that it can be used to guarantee the integrity of the data (e.g. to prevent the user from setting the month equal to 13). Java and C++ provide the keywords private, protected and public for data hiding.

Introduction to Object-Oriented Programming

- The whole idea behind encapsulation is to hide the implementation details from users. If a data member is private it means it can only be accessed within the same class. No outside class.

- How to implement encapsulation in java:
  1) Make the instance variables private so that they cannot be accessed directly from outside the class. You can only set and get values of these variables through the methods of the class.
  2) Have getter and setter methods in the class to set and get the values of the fields private data member (variable) of other class.

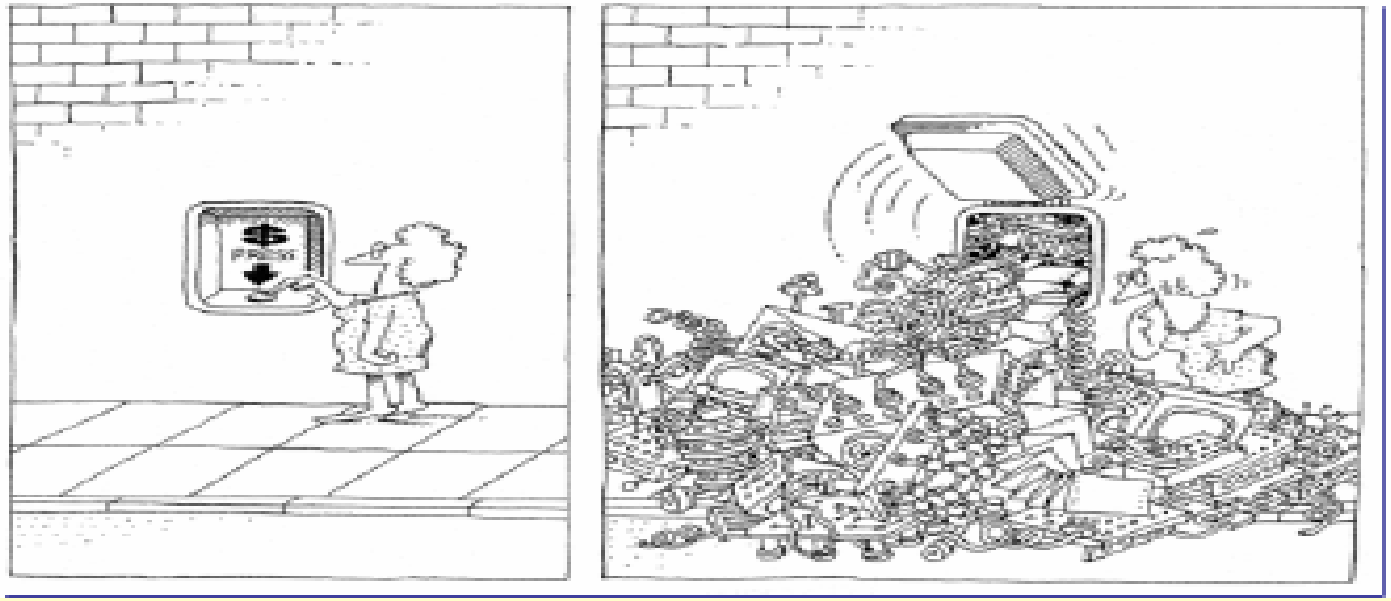
Data Encapsulation & Abstraction

1 – Abstract Specification

Abstraction

- Focus only on relevant properties of the problem
- 'Ignore' details.

Hiding internal details and showing functionality **is known as abstraction. For example phone call, we don't know the internal processing.**



Declaring a method with a parameter
A Simple Class

```
class Students  {
Private String name;

  public void dispalyName(String stdName)
{

    System.out.println(stdName+“Welcome to Class SW-Section-I”);
}  // end method

}  // end class
```

In the example, encapsulation is demonstrated as an OOP concept in Java. Here, the variable “name” is kept private or “encapsulated.”

Use Students Class

```
class StudentsTest  {


 public  static void main (String args []) {

// Creating an object  and asdign it to objStd

Students objStd=new Students();

{
// calling objStd’s method DisplayMsg()

objStd.displatname(“Ali");

}  // end main method

}  // end class
```

Declaring a class with Instance Variables
A Simple Class

```
class Students  {

String studentdame; // instance variable

  public void setName(String stdName)
{
       studentName=stdName;
}  // end method

 public String getName()
{
 return studentName;
}  // end method


}  // end class
```

Use Students Class

```
class StudentsTest  {


 public  static void main (String args []) {

// Creating an object  and asdign it to objStd

Students objStd=new Students();

{
// calling objStd's method DisplayMsg()

objStd.setName("Ali");
 System.out.println("Welcome"+ objStd.getName());

}  // end main method

}  // end class
```

Primitive Types vs. Reference Types

Data types in java are divided into two categories

Primitive Types and

Reference Types

Primitive Types are **byte, int, char, Boolean, short, long, float and double.**

A primitive variable can store exactly one value of its declared type at a time.

Reference Types

All non primitive types are references types, **so classes, we specify the types of objects are reference types**.

Variables of reference types **(normally called references)** store the **locations of objects in the computer's memory. Such variables are** said to refer to objects in the program.

```
Class Tea {

 float  milk;
 float  water ;
 float  sugar;
 float teadust;

 public void maketea(milk, water, sugar, teadust)
{
    // Code goes here
}
```