# Chapter Two

## Java Objects and classes


**Mulugeta G. and Sufian K.**

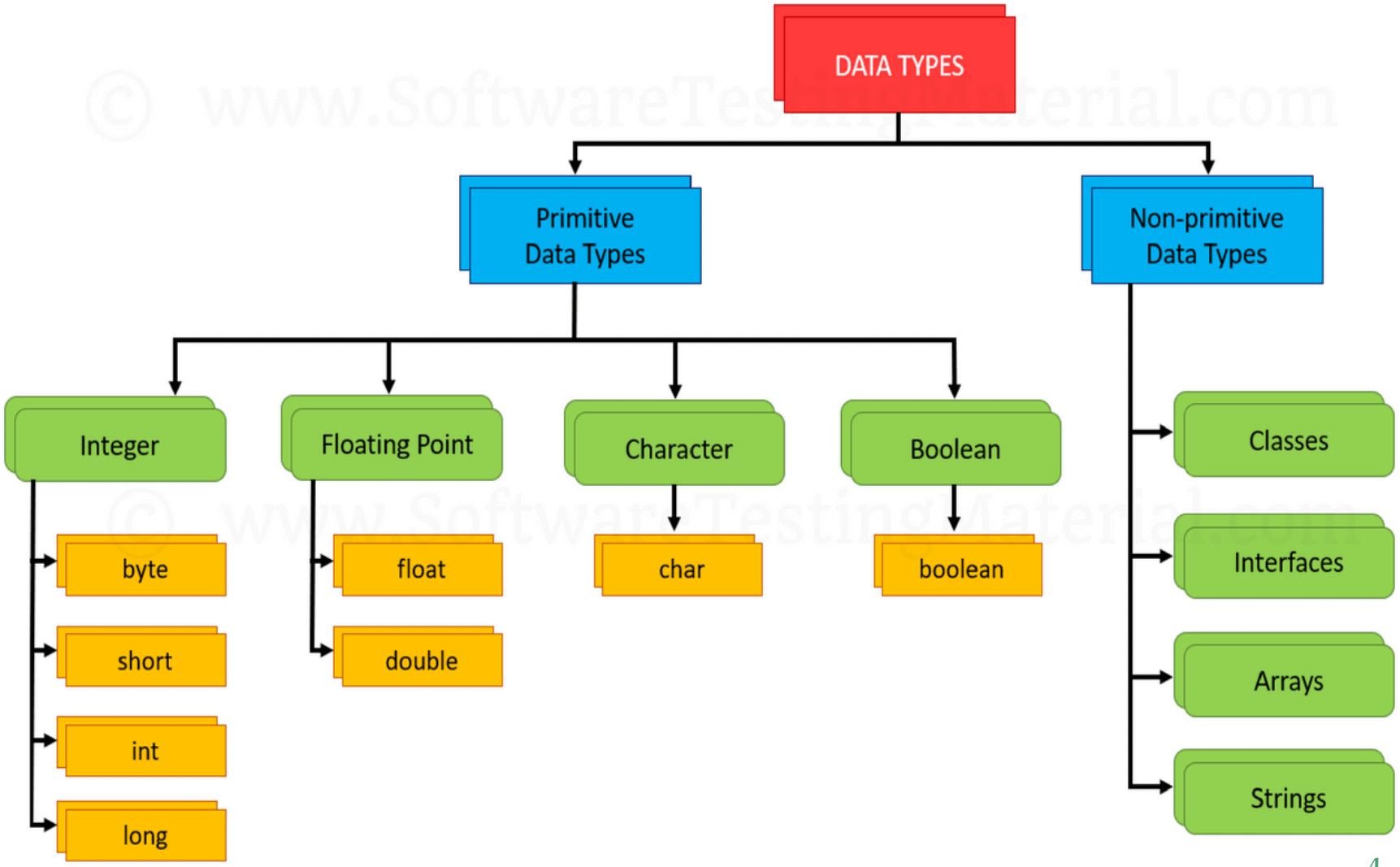
# Variables and Data Types

- **Variable**
  - A memory location with a name and a type that stores a value.
  - reserved memory locations to store values.
  - This means that when you create a variable you reserve some space in memory.
  - Their values can be changed at any point over the course of a program.
- Based on the data type of a variable, the operating system allocates memory and decides what can be stored in the reserved memory.
- Therefore, by assigning different data types to variables, you can store integers, decimals, or characters in these variables.

# Variables and Data Types

- **Data Types**
  - **Represent different values to be stored in the variable.**
  - **In java, there are two types of data types**
    - **Primitive Data Types**
    - **Non-Primitive Data Types**

# Variables and Data Types



# Primitive Datatype

- are predefined by the language and used as a Keywords.
- There are eight primitive data types supported by Java.

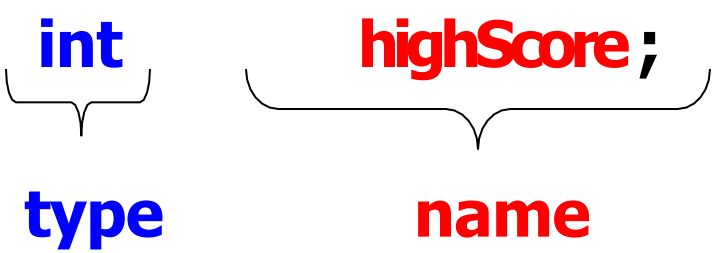
- byte
- short
- int
- long
- float
- double
- boolean
- char

# Non-primitive/Reference/Object Datatype

- Are created using defined constructors of the classes.
- They are used to access objects.
- These variables are declared to be of a specific type that cannot be changed.
- Default value of any reference variable is null.
- A reference variable can be used to refer to any object of the declared type or any compatible type.
- Example :
    - Arrays, Strings, Class, interface

# Variable Declaration

- You must declare all variables before they can be used
- The basic form of a variable declaration is:
  - Data-type variableName ;
- The type is listed first followed by the name.
- Example: a variable that stores an integer representing the highest score on an exam could be declared as follows:




- String studentName;
- boolean gameOver;

# Types of variables

- Variables are places where information can be stored while a program is running.
- There are three kind of variables in Java:
  - Local variables
  - Instance variables
  - Class/static variables

# Local Variable

- declared in methods, constructors, or blocks.
- created when the method, constructor or block is entered and the variable will be destroyed once it exits the method, constructor or block.
- Access modifiers cannot be used for local variables.
- visible only within the declared method, constructor or block.
- Local variables are implemented at stack level internally.
- There is no default value for local variables.
  - so local variablesshouldbe declaredand an initial value should be assigned before the first use.

# Instance Variable

- declared in a class, but outside a method, constructor or any block.
- created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.
- hold values that must be referenced by more than one method, constructor or block, or essential parts of an object's state that must be present throughout the class.
- Instance variables can be declared in class level before or after use.
- Access modifiers can be given for instance variables.

# Cont'd …

- **visible for all methods, constructors and block in the class.**
  - **Normally, it is recommended to make these variables private (access level). However visibility for subclasses can be given for these variables with the use of access modifiers.**
- **Instance variables have default values. For numbers the default value is 0, for Booleans it is false and for object references it is null.**
  - **Values can be assigned during the declaration or within the constructor.**
- **Instance variables can be accessed directly by calling the variable name inside the class. However within static methods and different class ( when instance variables are given accessibility) should be called using the fully qualified name.**
  - *ObjectReference.VariableName*.

# Class/static variables

- Class variables are also known as static variables
  - declared with the *static* keyword in a class, but outside a method, constructor or a block.
- rarely used other than being declared as constants.
  - Constants are variables that are declared as public/private, final and static.
  - Constant variables never change from their initial value.
- Static variables are stored in static memory. It is rare to use static variables other than declared final and used as either public or private constants.
- Static variables are created when the program starts, and destroyed when the program stops.

# Cont'd …

- most static variables are declared public since they must be available for users of the class.
- Default values are same as instance variables.
  - For numbers, the default value is 0; for Booleans, it is false; and for object references, it is null.
  - Values can be assigned during the declaration or within the constructor.
  - Additionally values can be assigned in special static initializer blocks.
- Static variables can be accessed by calling with the class name. *ClassName.VariableName*.
- When declaring class variables as public static final, then variables names (constants) are all in upper case.
  - If the static variables are not public and final the naming syntax is the same as instance and local variables.

# Objects

- A programming entity that contains state (data) and behavior (methods).
  - State: A set of values (internal data) stored in an object.
  - Behavior: A set of actions an object can perform, often reporting or modifying its internal state.
- Objects can be used as part of larger programs to solve problems.
- Example 1: Dogs
  - States: name, color, breed, and "is hungry?"
  - Behaviors: bark, run, and wag tail
- Example 2: Cars
  - States: color, model, speed, direction
  - Behaviors: accelerate, turn, change gears

# Class

- A class is the blueprint from which individual objects are created.
- Thus, a class is a template for an object, and an object is an instance of a class.
- A class is a collection of fields (data) and methods

```
class Dog {
        …
    description of a dog goes here
    …
}
```



# Constructing Objects

- We use the new keyword to construct a new instance of an Object.
- Syntax:

```
class_name object_name = new class_name();
```

**To Access Object's created:**

```
ObjectName.VariableName
ObjectName.MethodName(parameter-list)
```

- Example:

```
Dog d=new Dog();
```

- Know d is an object created form the Dog Class

# Constructor in Java

- Constructor is a special type of method that is used to initialize the object.
- Constructor is invoked at the time of object creation.
- Its main purpose is to set up the object's initial state.

- Rules for creating constructor
  - Constructor name must be same as its class name
    - Constructor Name=Class Name
  - Constructor do not have a return type

# Types of Constructor

- **Default constructor** (No-argument constructor)
  - A constructor that have no parameter
  - Default values to the object like 0, null etc. depending on the type.

```
<class_name>()
        {
        }
```

- **Parameterized constructor**
  - A constructor that have parameters
  - Used to provide different values to the distinct objects.

```
<class_name>(parameter list)
        {
        }
```

# Example

```
class Student {
    int id;
    String name;
void display()
        {
System.out.println(id+" "+name);
        }
public static void  main(String args[]) {
    Student s1=new Student();
    Student s2=new Student();
    s1.display();
    s2.display();
        }
     }
```

class_name object_name=new class_name();

# Example:

```
class Student
{
int id;
String name;
Student(int i,String n){
id = i;
name = n;
}

void display(){
System.out.println(id+" "+na
me);
}
public static void main(String args[])
{
Student s1= new Student(111,"Abay");

Student s2 = new Student(222,"Gebissa");

s1.display();
s2.display();
}
}
```

## Constructor Overloading

- Constructor overloading is a technique in Java in which a class can have any number of constructors that differ in parameter lists.
- The compiler differentiates these constructors by taking into account the number of parameters in the list and their type.

## Method Overloading

- class have multiple methods by the same name but different parameters

- Method Overloading

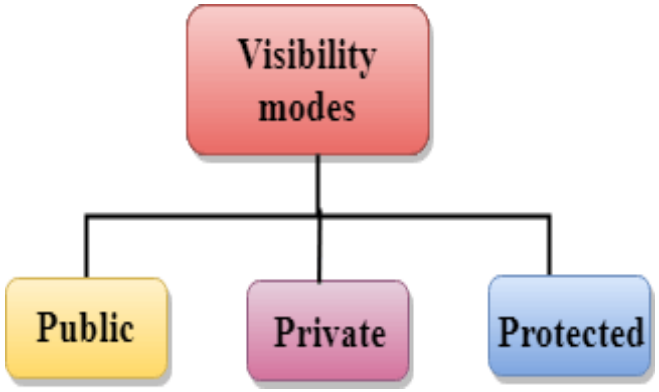
```
class Calculation{
void sum(int a,int b)
{
System.out.println(a+b);
}
void sum(int a,int b,int c)
{
System.out.println(a+b+c);
}
public static void main(String args[])
{
Calculation obj=new Calculation();
obj.sum(10,10,10);
obj.sum(20,20);
}}
```

# Difference between Constructor Vs Method

| Constructor | Method |
| --- | --- |
| • Constructor is used to initialize the state of an object. | • Method is used to expose behaviour of an object. |
| • Constructor must not have return type. | • Method must have return type. |
| • Constructor is invoked implicitly. | • Method is invoked explicitly. |
| • The java compiler provides a default constructor if you don't have any constructor. | • Method is not provided by compiler in any case. |
| • Constructor name must be same as the class name. | • Method name may or may not be same as class name. |

# Class Member Visibility

- Class member visibility controls which parts of code can access a class's members.
- define the scope of a class member.
  - **Private**: Only accessible from other members of the same class
  - **Protected**: Accessible from the class itself and by inheriting and parent classes
  - **Public**: Accessible from anywhere



# Methods

- **A method is a block of code which only runs when it is called.**
- **You can pass data, known as parameters, into a method.**
- **Methods are used to perform certain actions, and they are also known as functions.**
- **Why use methods?**
  - **To reuse code:**
  - **Define the code once, and use it many times.**

**Create a Method**

- **A method must be declared within a class.**
- **It is defined with the name of the method, followed by parentheses ().**
- **Java provides some pre-defined methods, such as System.out.println**

# Thank You