CSE 1203

Object Oriented Programming [C++]

Chapter 5:
Java Programming -01

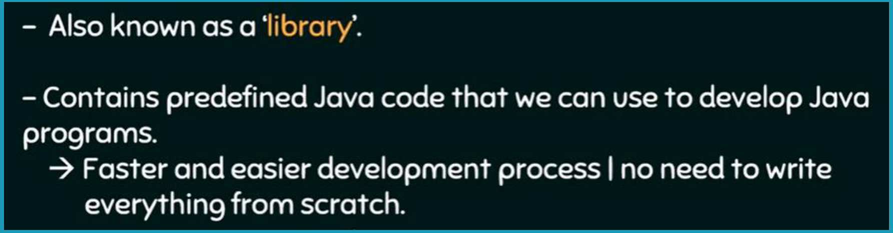
Java: Introduction

What is Java?

Java is a popular object oriented programming language, created in 1995.
It is owned by Oracle, and more than **3 billion** devices run Java.
It is used for:

- Mobile applications (specially Android apps)
- Desktop applications
- Web applications
- Web servers and application servers
- Games
- Database connection
- And much, much more!

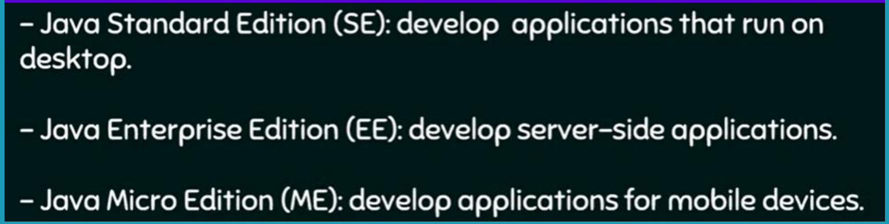
API: Application Programming Interface



- Also known as a 'library'.

- Contains predefined Java code that we can use to develop Java programs.
  → Faster and easier development process | no need to write everything from scratch.

Java Editions: 3 types

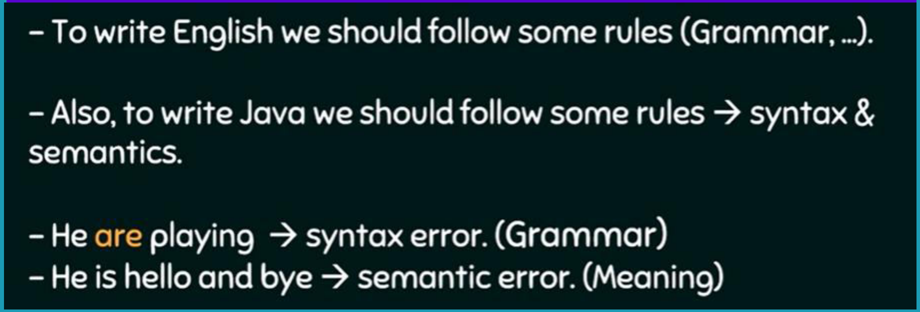


- Java Standard Edition (SE): develop applications that run on desktop.

- Java Enterprise Edition (EE): develop server-side applications.

- Java Micro Edition (ME): develop applications for mobile devices.

Java Specification: java syntax & semantic

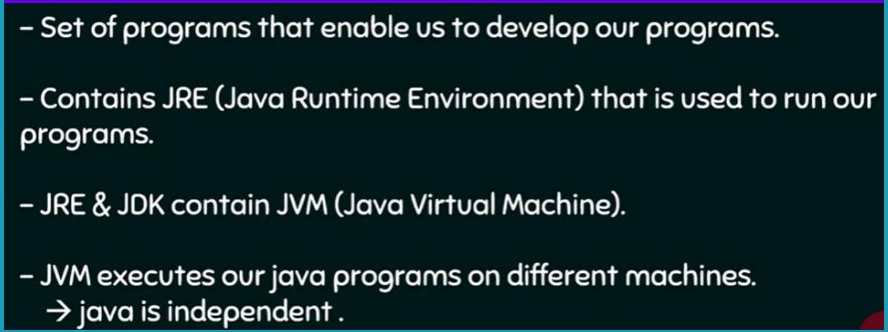


- To write English we should follow some rules (Grammar, ...).

- Also, to write Java we should follow some rules → syntax & semantics.

- He are playing → syntax error. (Grammar)
- He is hello and bye → semantic error. (Meaning)

JDK: Java Development Kit



- Set of programs that enable us to develop our programs.

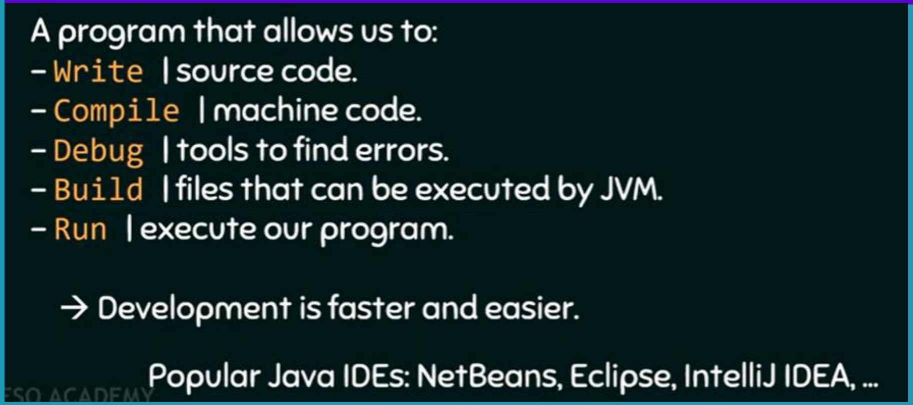
- Contains JRE (Java Runtime Environment) that is used to run our programs.

- JRE & JDK contain JVM (Java Virtual Machine).

- JVM executes our java programs on different machines.
  → java is independent .

Java: Introduction

IDE: Integrated Development Environment

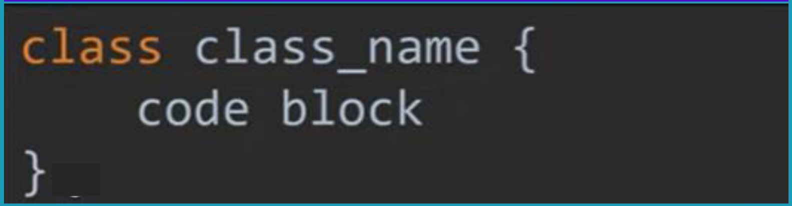


A program that allows us to:
- Write | source code.
- Compile | machine code.
- Debug | tools to find errors.
- Build | files that can be executed by JVM.
- Run | execute our program.

→ Development is faster and easier.

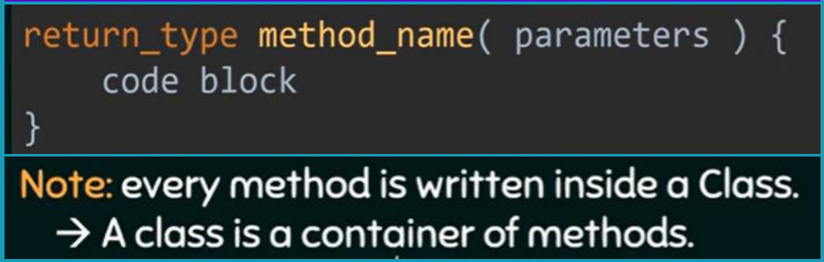
Popular Java IDEs: NetBeans, Eclipse, IntelliJ IDEA, ...

Java: Class Structure



```
class class_name {
    code block
}
```

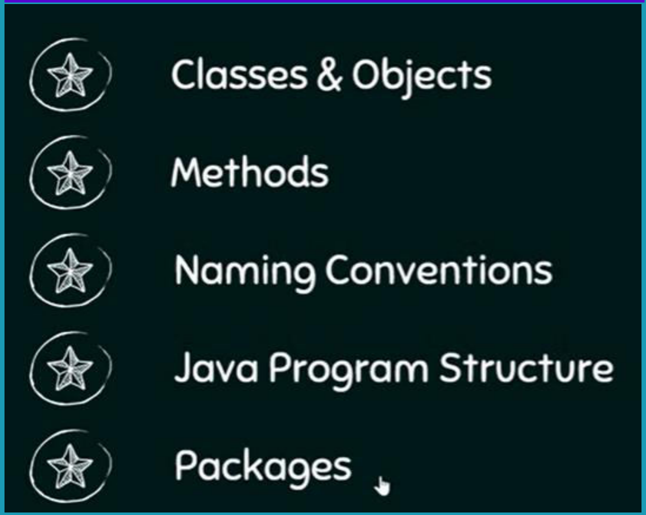
Java : Methods



```
return_type method_name( parameters ) {
    code block
}
```

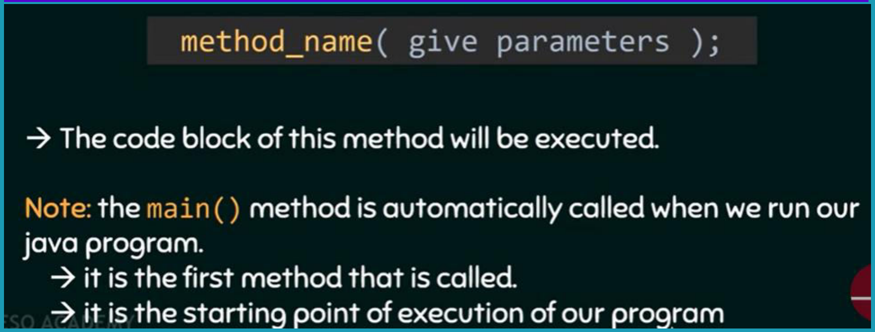
Note: every method is written inside a Class.
→ A class is a container of methods.

Java: Basic Concepts



- Classes & Objects
- Methods
- Naming Conventions
- Java Program Structure
- Packages

Java : Method Calling



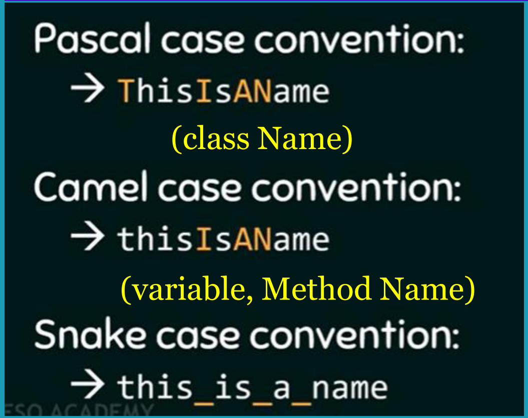
```
method_name( give parameters );
```

→ The code block of this method will be executed.

Note: the main() method is automatically called when we run our java program.
→ it is the first method that is called.
→ it is the starting point of execution of our program

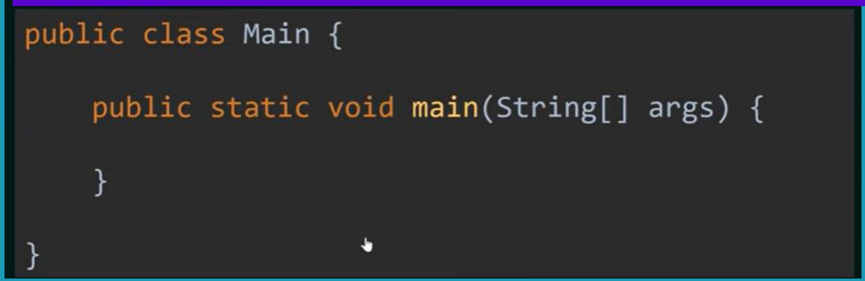
Java: Introduction

Java: Naming Convention



Pascal case convention:
→ ThisIsAName
(class Name)
Camel case convention:
→ thisIsAName
(variable, Method Name)
Snake case convention:
→ this_is_a_name

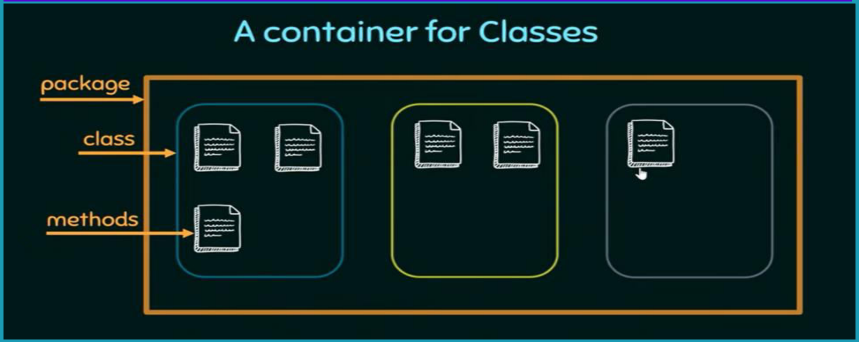
Java: Programming Structure



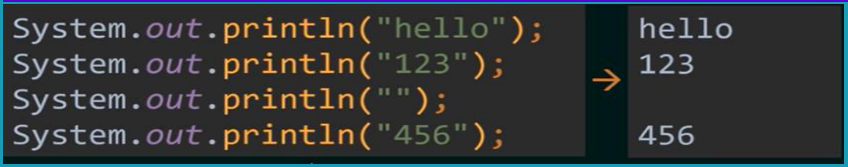
```
public class Main {

    public static void main(String[] args) {

    }

}
```

Note: Each java program must have a class that contain **`main()`** method

Java : Package

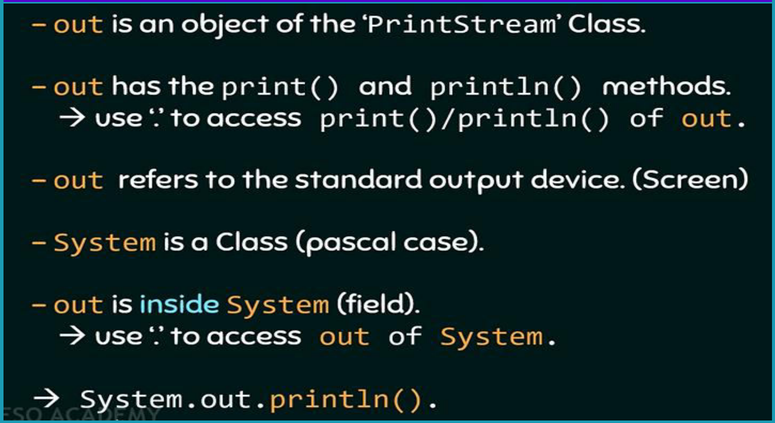


Java: println() method



```
System.out.println("hello");          hello
System.out.println("123");       →    123
System.out.println("");
System.out.println("456");            456
```

Java : System.out

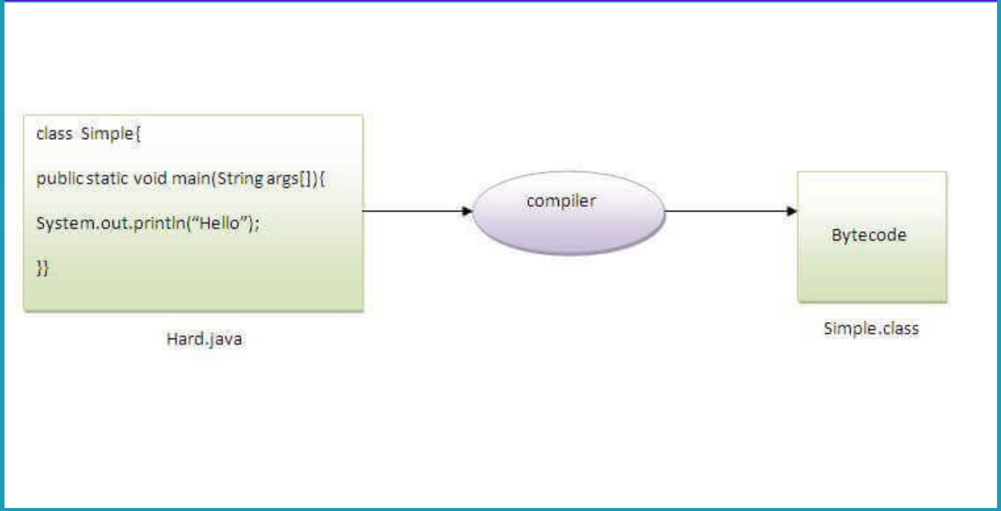


- out is an object of the 'PrintStream' Class.
- out has the print() and println() methods.
  → use '.' to access print()/println() of out.
- out refers to the standard output device. (Screen)
- System is a Class (pascal case).
- out is inside System (field).
  → use '.' to access out of System.

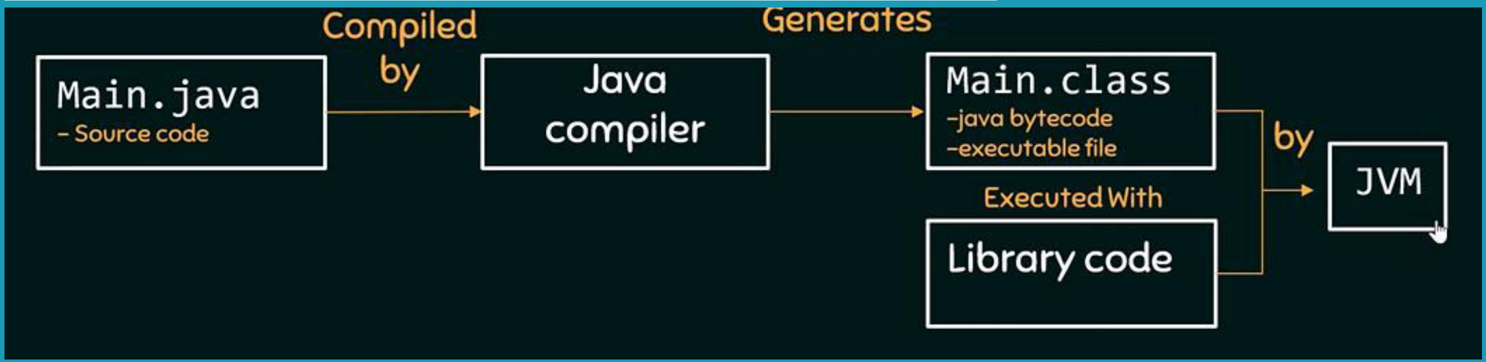
→ System.out.println().

Java: Introduction

Java: Compiling



class Simple{
public static void main(String args[]){
System.out.println("Hello");
}}
Hard.java
compiler
Bytecode
Simple.class



Java Virtual Machine, or JVM, loads, verifies and executes Java bytecode. It is known as the interpreter or the core of Java programming language because it executes Java programming.

Java: Program

Program 1: print a message/text

```
package CSE1203;

public class First {
 public static void main(String[] args) {
  System.out.print("Welcome to Java World");
 }
}
```

Output

```
Welcome to Java World
```

Notes: The class name must be the same as the .java source filename and package should be declared at first

Program 2: print a message/text

```
package CSE1203;
public class Second {
 public static void main(String[] args) {
  System.out.print("Talk less listen more");
 }
}
```

Notes: A class Second is created under package CSE1203

Program 2 (ex): call main() of second

```
package CSE1203;
public class First {
 public static void main(String[] args) {
 System.out.println("Welcome to Java World");
 Second.main(null);
 }
}
```

Output

```
Welcome to Java World
Talk less listen more
```

Notes: In the First class just write the class name if you want to call other classes method

Java: Program

Program 2 (ex): change Second class

```
package CSE1203;
public class Second {
 public static void main(String[] args) {
  System.out.println("Talk less listen more");
  Display();
}
public static void Display() {
 System.out.println("Honesty is the best policy");
}
}
```

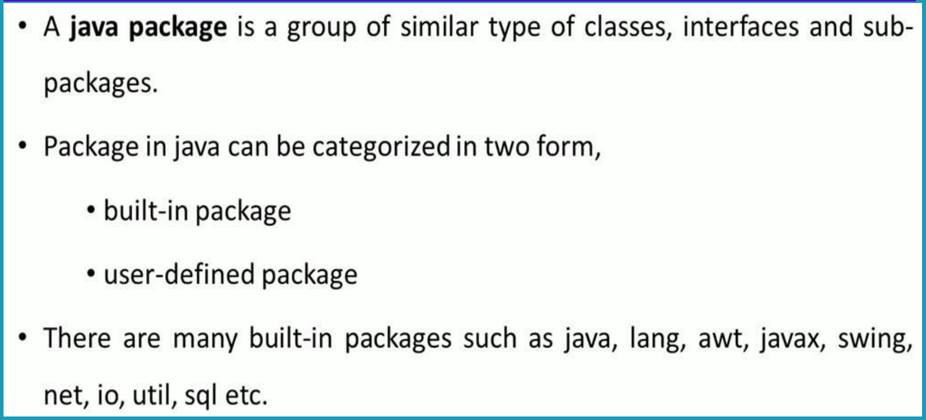
Output

```
Welcome to Java World
Talk less listen more
Honesty is the best policy
```

Notes: From First class, main() of Second class is called and inside the main() of Second, Display method is called

There are no restrictions on the number of classes that can be present in one Java program. But each Java program should have only one class declared with public access specifier. There cannot be two public classes in a single Java program.

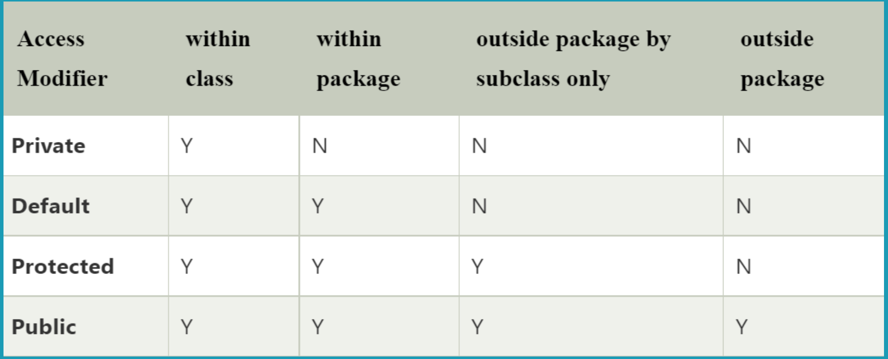
package in Java



- A **java package** is a group of similar type of classes, interfaces and sub-packages.
- Package in java can be categorized in two form,
  - built-in package
  - user-defined package
- There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

Access Modifiers

```
1.  public  :  access from everywhere
2.  private:  access only inside the class
3.  protected:  access from everywhere
4.  default:  access from everywhere
```



Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

Java: Program

Program 3: private method

```
package CSE1203;
public class Second {
 public static void main(String[] args) {
  System.out.println("Talk less listen more");
  Display();
}
private static void Display() {
 System.out.println("Honesty is the best policy");
}
}
```

Output

```
Talk less listen more
Honesty is the best policy
```

Notes: It is run from Second but in the following it is run from First. It produces error as Display() is private to Second class

```
package CSE1203;
public class First {
 public static void main(String[] args) {
 System.out.println("Welcome to Java World");
 Second.Display());
  }
}
```

Program 4: static method

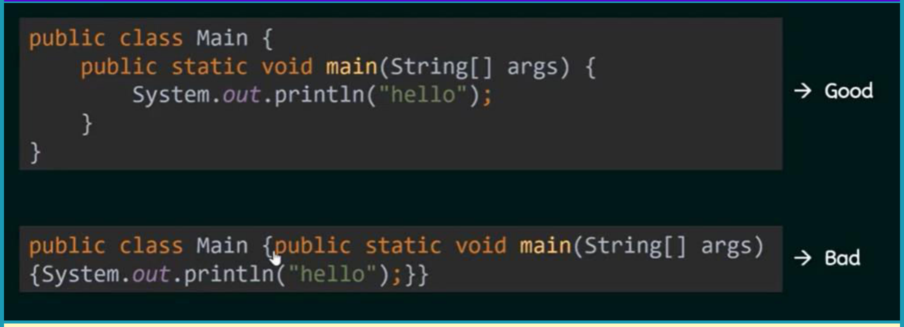
```
package CSE1203;
public class Second {
 public static void main(String[] args) {
  System.out.println("Talk less listen more");
  Second.Display();
}
private static void Display() {
 System.out.println("Honesty is the best policy");
}
}
```

Notes: static method can be called by its class name with (.) operator. But if you remove the static from Display() in the following then it can not be called from main(). Remember that a non-static method can not be called from a static method

```
package CSE1203;
public class Second {
 public static void main(String[] args) {
  System.out.println("Talk less listen more");
  Display(); //produces error
}
Private void Display() {
 System.out.println("Honesty is the best policy");
}
}
```

Java: Program

Programming Style



```
public class Main {
    public static void main(String[] args) {
        System.out.println("hello");
    }
}
```

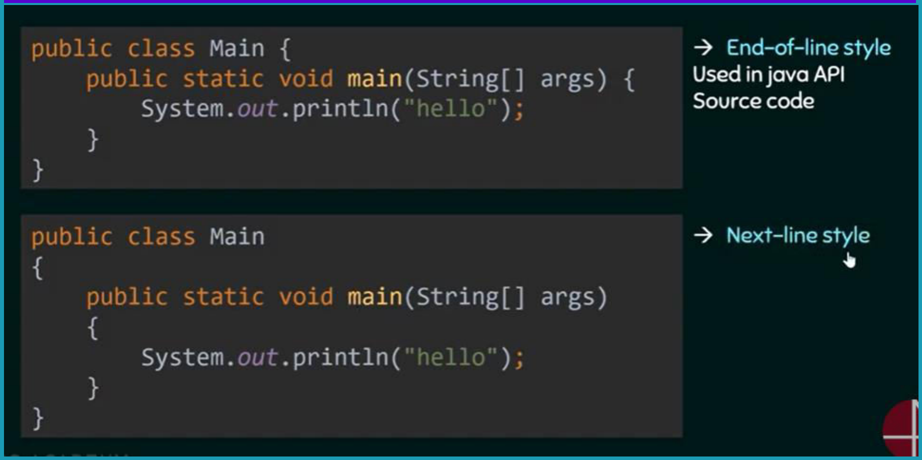
→ Good

```
public class Main {public static void main(String[] args)
{System.out.println("hello");}}
```

→ Bad

Notes: Use proper spacing and indentation

Programming Style : block



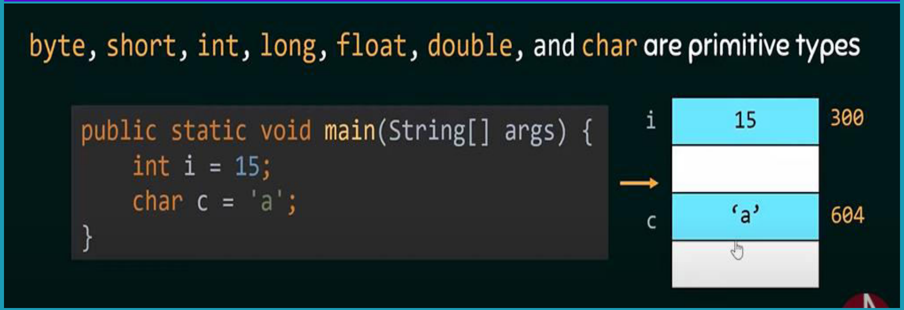
```
public class Main {
    public static void main(String[] args) {
        System.out.println("hello");
    }
}
```

→ End-of-line style
Used in java API
Source code

```
public class Main
{
    public static void main(String[] args)
    {
        System.out.println("hello");
    }
}
```

→ Next-line style

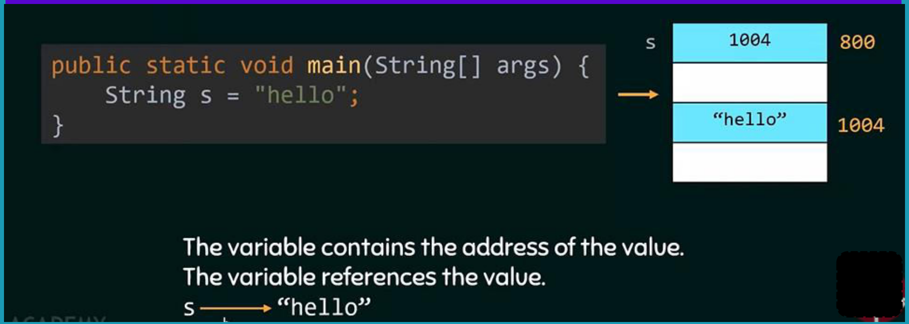
Data Types: primitives



byte, short, int, long, float, double, and char are primitive types

```
public static void main(String[] args) {
    int i = 15;
    char c = 'a';
}
```

Data Types: reference



```
public static void main(String[] args) {
    String s = "hello";
}
```

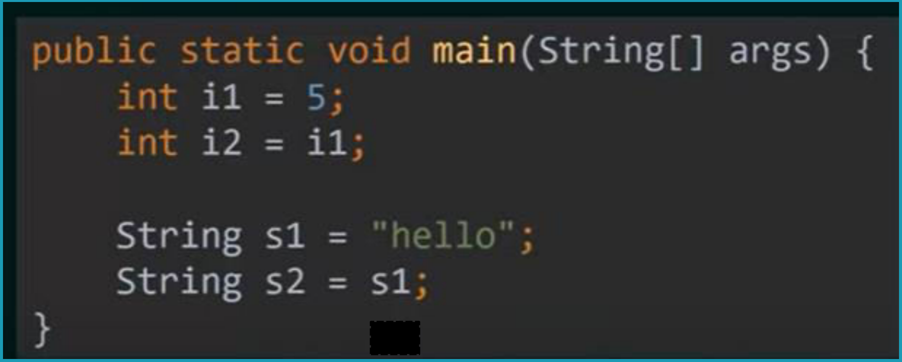
The variable contains the address of the value.
The variable references the value.
s ⟶ "hello"

string class: few methods

charAt()	isEmpty()	toLowerCase()
compareTo()	length()	toUpperCase()
concat()	replace()	toString()
equals()	substring()	trim()

Java: Program

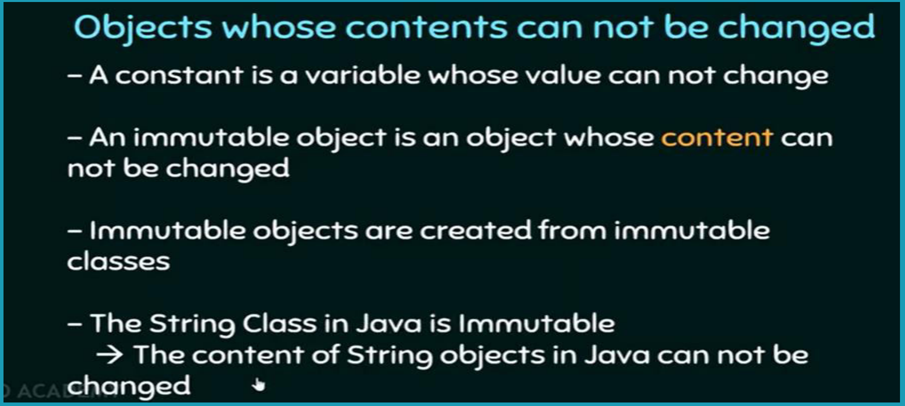
Primitive vs Reference Types



```
public static void main(String[] args) {
    int i1 = 5;
    int i2 = i1;

    String s1 = "hello";
    String s2 = s1;
}
```

[redacted]

Immutable Objects: primitives



Objects whose contents can not be changed

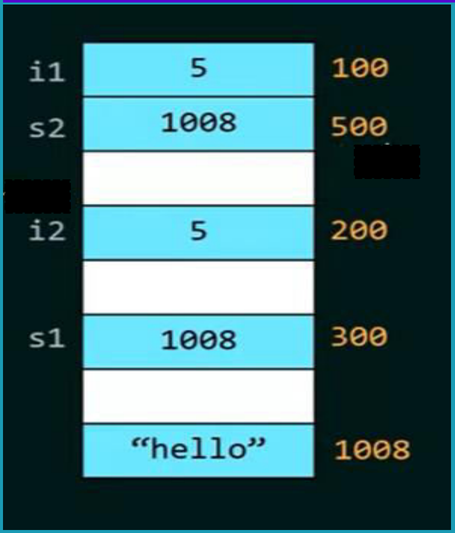
– A constant is a variable whose value can not change

– An immutable object is an object whose content can not be changed

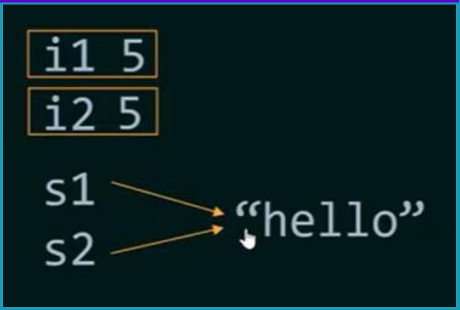
– Immutable objects are created from immutable classes

– The String Class in Java is Immutable
  → The content of String objects in Java can not be changed

Memory Allocation



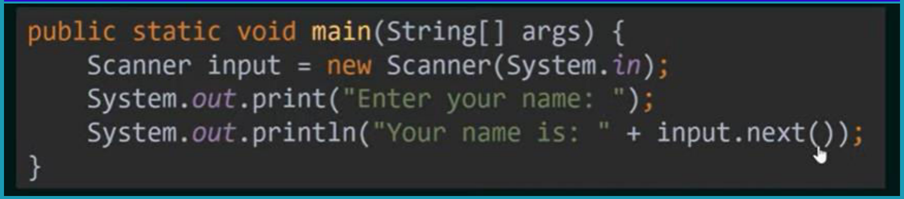
[redacted]



String class creates objects in String Constant Pool (SCP) and only one object is created for same literal

Java: Program

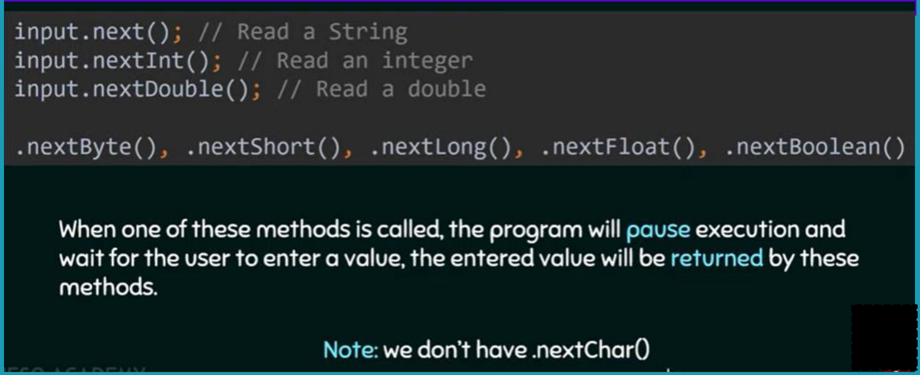
scanner class : next() method



```
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.print("Enter your name: ");
    System.out.println("Your name is: " + input.next());
}
```

```
Kamal
Your name is Kamal
```

Methods: scanner class



```
input.next(); // Read a String
input.nextInt(); // Read an integer
input.nextDouble(); // Read a double

.nextByte(), .nextShort(), .nextLong(), .nextFloat(), .nextBoolean()
```

When one of these methods is called, the program will pause execution and wait for the user to enter a value, the entered value will be returned by these methods.

Note: we don't have .nextChar()

scanner class : nextInt() method

```
package CSE1203;

import java.util.Scanner;

public class First {

public static void main(String[] args) {
Scanner scan=new Scanner(System.in);
int a=scan.nextInt();
int b=scan.nextInt();
int s=a+b;
System.out.println("sum="+s);
}
}
```

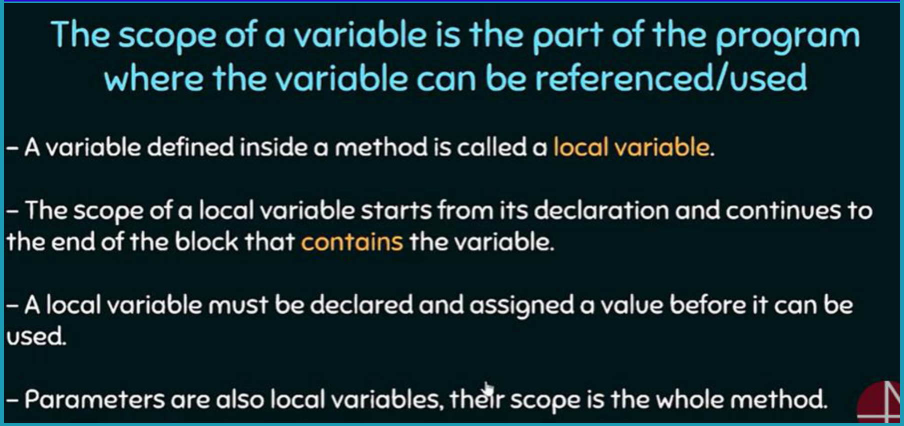
Here two integers input from keyboard stored in variable a and b. Then s stored sum of a & b

Built in Statements

`if, switch, for, while,do-while` are the same as C/C++l

Java: Program

Local Variable and its Scope



The scope of a variable is the part of the program where the variable can be referenced/used

– A variable defined inside a method is called a local variable.

– The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable.

– A local variable must be declared and assigned a value before it can be used.

– Parameters are also local variables, their scope is the whole method.

Java: array declaration

```
dataType[] arr; (or)
dataType []arr; (or)
dataType arr[];
```

Java: array instantiation

```
arrayRefVar=new datatype[size];
```

Example: Local Variable

```
package CSE1203;
public class First {

public static void main(String[] args) {
 if(true) {
   int i=5;
   System.out.println("i="+i);
  }
 System.out.println("i="+i);
}
```

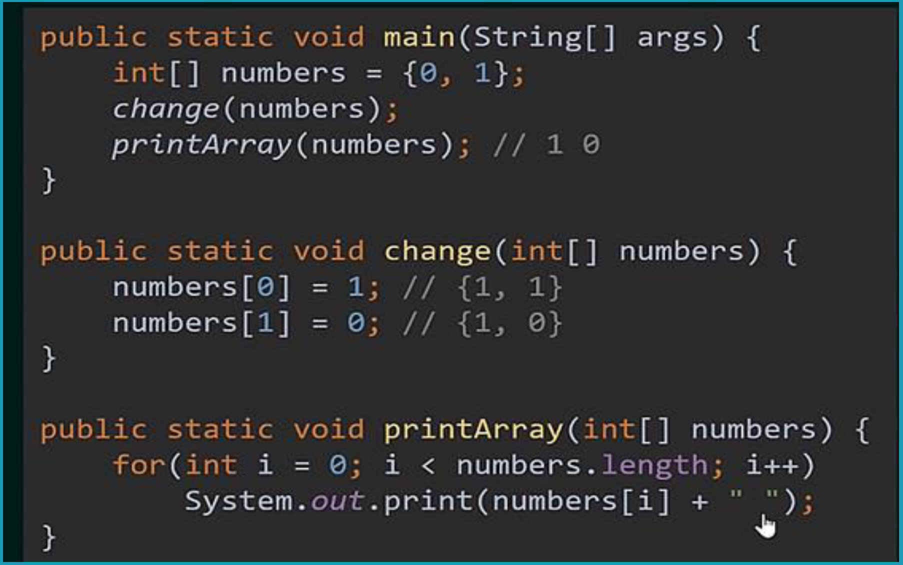
Here error occurs at second println as i declared inside block if

Example: array

```
package CSE1203;
public class First {

public static void main(String[] args) {
int[]  ax=new int[5];
ax[0]=10; ax[1]=20;
for(int i=0;i<5;i++)
 System.out.println(" "+ax[i]);
}
}
```

Java: Program

Example: array pass by reference



```
public static void main(String[] args) {
    int[] numbers = {0, 1};
    change(numbers);
    printArray(numbers); // 1 0
}

public static void change(int[] numbers) {
    numbers[0] = 1; // {1, 1}
    numbers[1] = 0; // {1, 0}
}

public static void printArray(int[] numbers) {
    for(int i = 0; i < numbers.length; i++)
        System.out.print(numbers[i] + " ");
}
```

Here array is an object and it is passed by reference

Java: anonymous object

Anonymous means **Nameless**. An anonymous object is basically a value that has been created but has no name.

Example: anonymous object

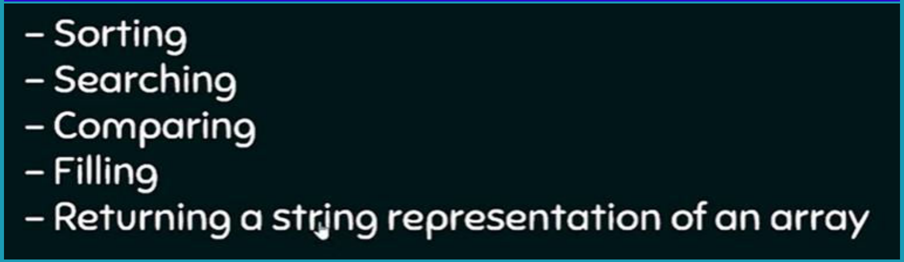
```
package CSE1203;
import java.awt.Point;
public class First {
 public static void main(String[] args) {
  Point p=getPoint();
  System.out.println("p="+p);
 }
 private static Point getPoint() {
 return new Point(1,2); //annonymous object
 }
}
```

Java: array of anonymous object

```
package CSE1203;
import java.awt.Point;
public class First {
 public static void main(String[] args) {
  int[] numbers=getNumber();
  printArray(numbers);
}
 public static void printArray(int[] numbers) {
   for(int i=0;i<numbers.length;i++)
   System.out.print(" "+numbers[i]);
}
 public static int[] getNumber() {
     return new int[] {1,2,3,4,5};
}
}
```

Java: Program

Java: Array class methods for



- Sorting
- Searching
- Comparing
- Filling
- Returning a string representation of an array

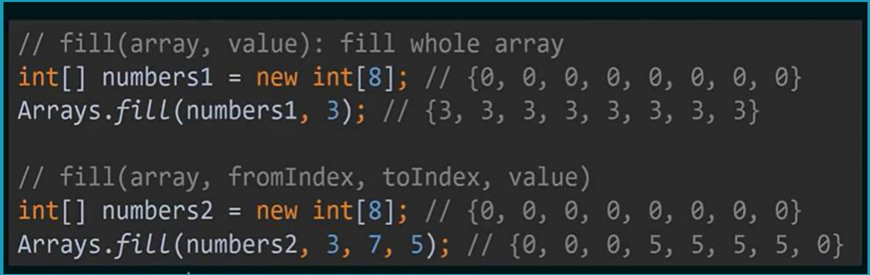
Java: Arrays sort() method

```
package CSE1203;
import java.util.Arrays;

public class First {
 public static void main(String[] args) {
  int[] numbers= {10,20,70,90,30,80};
  System.out.println("Initial Array:");
  printArray(numbers);
  Arrays.sort(numbers);
  System.out.println("\nSorted Array:");
  printArray(numbers);
}
 private static void printArray(int[] numbers) {
  for(int i=0;i<numbers.length;i++)
    System.out.print(" "+numbers[i]);
}
}
```

Java: Arrays BinarySearch() method

```
package CSE1203;
import java.util.Arrays;
public class First {
 public static void main(String[] args) {
  int[] numbers= {10,20,70,90,30,80};
  System.out.println("Initial Array:");
  printArray(numbers);
  Arrays.sort(numbers);
  System.out.println("\nSorted Array:");
  printArray(numbers);
  int i=Arrays.binarySearch(numbers, 44);
  //returns index or if not found returns -ve number
  System.out.println("\nIndex of found element="+i);
}
 private static void printArray(int[] numbers) {
  for(int i=0;i<numbers.length;i++)
    System.out.print(" "+numbers[i]);
}
}
```

Java: Arrays fill() method



```
// fill(array, value): fill whole array
int[] numbers1 = new int[8]; // {0, 0, 0, 0, 0, 0, 0, 0}
Arrays.fill(numbers1, 3); // {3, 3, 3, 3, 3, 3, 3, 3}

// fill(array, fromIndex, toIndex, value)
int[] numbers2 = new int[8]; // {0, 0, 0, 0, 0, 0, 0, 0}
Arrays.fill(numbers2, 3, 7, 5); // {0, 0, 0, 5, 5, 5, 5, 0}
```

Java: Program

Java: Arrays toString() method

```java
package CSE1203;
import java.util.Arrays;
public class First {
 public static void main(String[] args) {
  int[] ax= {10,20,70,90,30,80};
  System.out.println("Initial Array:");
  System.out.println(Arrays.toString(ax));
}
}
```

Output

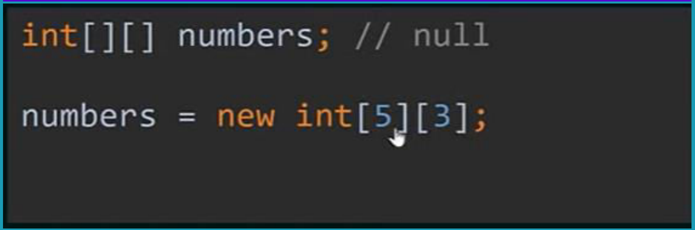
```
Initial Array:
[10, 20, 70, 90, 30, 80]
```

Note: The **`toString()`** method returns the String representation of the object. By overriding the **`toString()`** method of the Object class, we can return values of the object, so we don't need to write much code.

Java: Variable Length Parameter(…)

```java
package CSE1203;
import java.util.Arrays;
public class First {
 public static void main(String[] args) {
   int[] ax= {10,20,70,90,30,80};
   System.out.println(sum(1,2,3));
   System.out.println(sum(1,2,3,4));
   System.out.println(sum(ax));
}
 public static int sum(int...ax) {
   int s=0;
   for(int i=0;i<ax.length;i++)
   s=s+ax[i];
   return s;
}
}
```

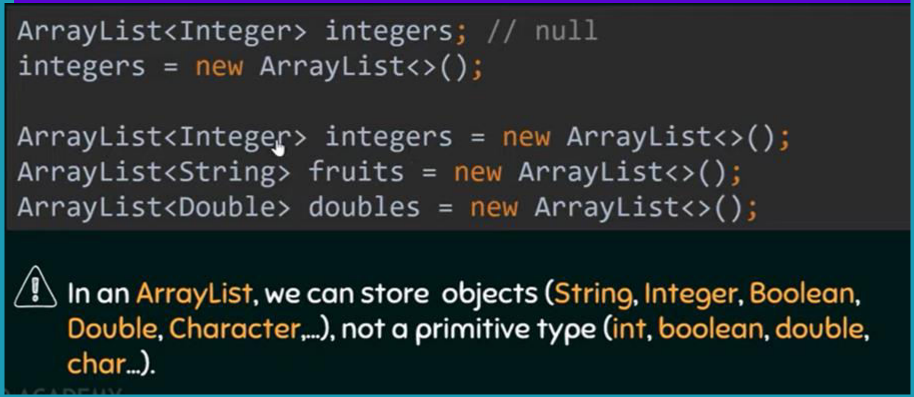
Java: 2D array Declaration



```java
int[][] numbers; // null

numbers = new int[5][3];
```

Java: Program

Java: ArrayList class

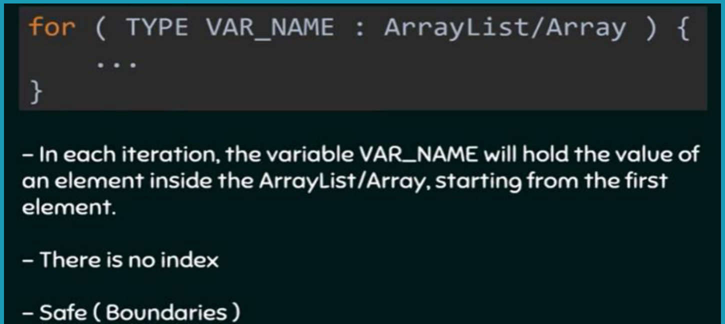


```
ArrayList<Integer> integers; // null
integers = new ArrayList<>();

ArrayList<Integer> integers = new ArrayList<>();
ArrayList<String> fruits = new ArrayList<>();
ArrayList<Double> doubles = new ArrayList<>();
```

⚠ In an ArrayList, we can store objects (String, Integer, Boolean, Double, Character,...), not a primitive type (int, boolean, double, char...).

Note: The **`ArrayList`** class is a resizable/ variable length array. It includes methods **`add()`, `set()`, `remove()`, `size()`, `clear()`** etc.

Note: To display the content of an array, it should be written inside the print() method. To write `toString()` method after array object is not mandatory



```
for ( TYPE VAR_NAME : ArrayList/Array ) {
    ...
}
```

– In each iteration, the variable VAR_NAME will hold the value of an element inside the ArrayList/Array, starting from the first element.

– There is no index

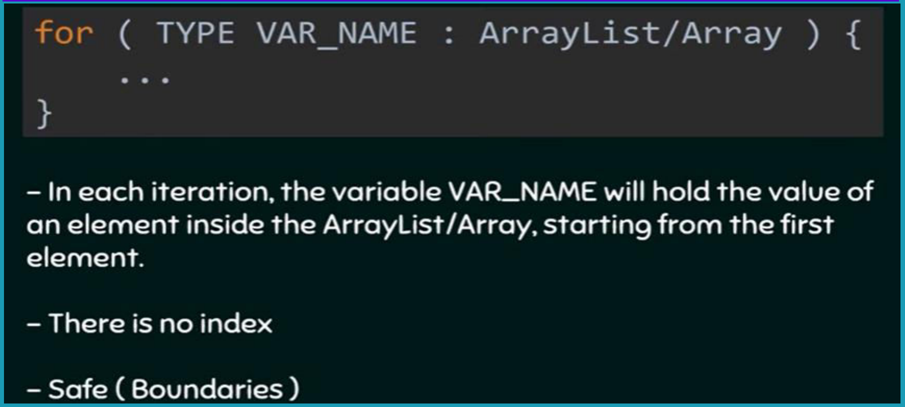
– Safe (Boundaries)

Example: ArrayList class

```
package CSE1203;
import java.util.ArrayList;
import java.util.Arrays;
public class First {
public static void main(String[] args) {
ArrayList<String> fruits=new ArrayList<>();
fruits.add("Apple");//insert at back
fruits.add("Mango");//insert at back
fruits.add("Orange");//insert at back
System.out.println(fruits);
fruits.add(0,"Banana");//insert at index 0
System.out.println(fruits);
System.out.println(fruits.get(0)); //get element
of index 0
fruits.set(1, "Guava"); //change value at index 1
System.out.println(fruits);
fruits.remove(2); //delete value at index 2
System.out.println(fruits);
fruits.remove("Orange"); //delete by value
System.out.println(fruits.toString()); //same
System.out.println(fruits.size()); //Total
elements
fruits.sort(null); //sort elements
Collections.sort(fruits);  //Alternative sort
fruits.clear(); //delete all elements
System.out.println(fruits);
}
}
```

Java: Program

Java: for-each loop



```
for ( TYPE VAR_NAME : ArrayList/Array ) {
    ...
}
```

– In each iteration, the variable VAR_NAME will hold the value of an element inside the ArrayList/Array, starting from the first element.

– There is no index

– Safe (Boundaries)

Example: for-each loop

```
package CSE1203;
import java.util.ArrayList;
import java.util.Arrays;

public class First {
 public static void main(String[] args) {
  ArrayList<String> fruits=new ArrayList<>();
  fruits.add("Orange");//insert at back
  fruits.add("Mango");//insert at back
  fruits.add("Apple");//insert at back
  for(String i:fruits)
  System.out.print(" "+i);
 }
}
```

Java: class & object

Java: class-object

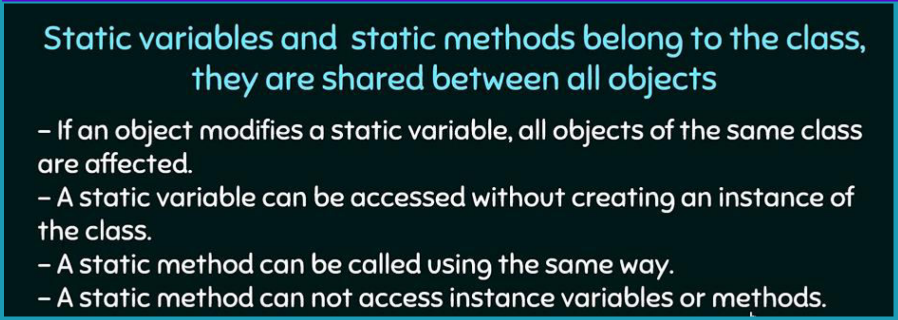
```java
package CSE1203;
import java.awt.Point;

public class First {

public static void main(String[] args) {
  Circle c1=new Circle(new Point(4,2),3);
  System.out.println("Center="+c1.getCenter());
  System.out.println("Area="+c1.Area());
}
}
```

Note: here new Point(4,2) is anonymous object. Class Circle can be define in the same file with main() or in the different file.

```java
package CSE1203;

import java.awt.Point;

class Circle{
Point c;
int r;
public
Circle(Point c,int r){
 this.c=c;
 this.r=r;
}
double Area() {
return 2*3.14*r;
}
Point getCenter() {
return c;
}
}
```

Java: static variable/method

Java: static variable/method



Static variables and static methods belong to the class, they are shared between all objects

- If an object modifies a static variable, all objects of the same class are affected.
- A static variable can be accessed without creating an instance of the class.
- A static method can be called using the same way.
- A static method can not access instance variables or methods.

Java: static variable/method

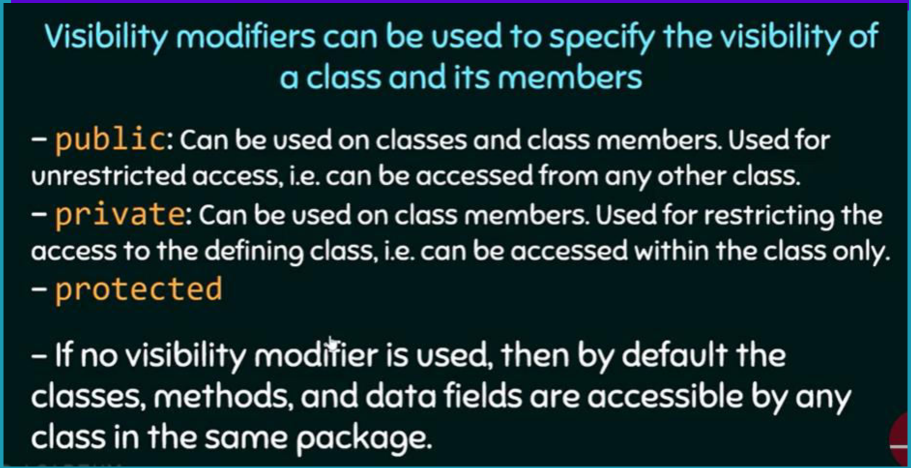
```
package CSE1203;
import java.awt.Point;

public class First {

public static void main(String[] args) {
Circle c1=new Circle(new Point(4,2),3);
Circle c2=new Circle(new Point(1,2),4);
System.out.println("Center="+c1.getCenter());
System.out.println("Area="+c1.Area());
System.out.println("Circle
Count="+c1.getCount());
}
}
```

```
package CSE1203;
import java.awt.Point;

class Circle{
Point c;
int r;
static int count=0;
public
Circle(Point c,int r){
 this.c=c;
 this.r=r;
 count++;
}
double Area() {
return 2*3.14*r;
}
Point getCenter() {
return c;
}
static int getCount() {
return count;
}
}
```

Note: here count is a static variable and it is common for all objects. To return a static variable a static method is used.

Java: static variable/method

Java: Visibility Modifiers

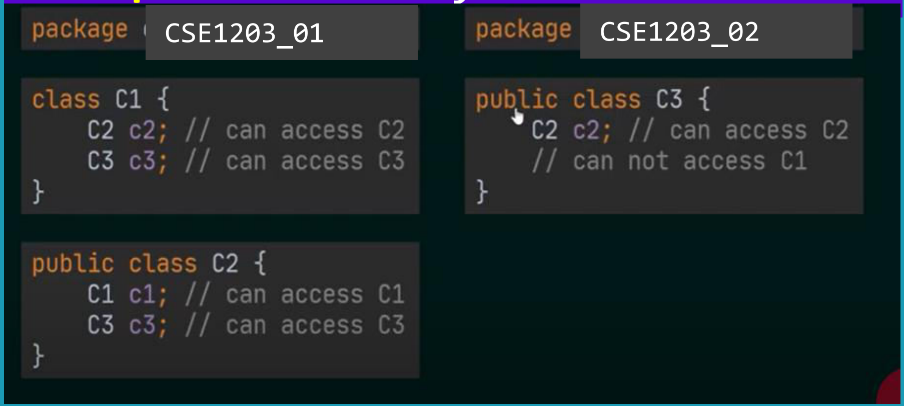


Visibility modifiers can be used to specify the visibility of a class and its members

- public: Can be used on classes and class members. Used for unrestricted access, i.e. can be accessed from any other class.
- private: Can be used on class members. Used for restricting the access to the defining class, i.e. can be accessed within the class only.
- protected

- If no visibility modifier is used, then by default the classes, methods, and data fields are accessible by any class in the same package.

Example: Visibility Modifiers



```
package CSE1203_01

class C1 {
    C2 c2; // can access C2
    C3 c3; // can access C3
}

public class C2 {
    C1 c1; // can access C1
    C3 c3; // can access C3
}
```

```
package CSE1203_02

public class C3 {
    C2 c2; // can access C2
    // can not access C1
}
```

Note: public class can be access from any package but default class can be accessed within the package

Java: Inner class

In Java, inner class refers to **the class that is declared inside class or interface.**Inner classes are **a security mechanism in Java**. We know a class cannot be associated with the access modifier private, but if we have the class as a member of other class, then the inner class can be made private. And this is also used to access the private members of a class.

```
package CSE1203;
public class First {
public static void main(String[] args) {
  //outer is Outer class object
  Outer outer=new Outer();
  outer.Display();
  //in is Inner class object
  Outer.Inner in = new Outer().new Inner(45);
  in.getY();
}
}
```

Note: Here object of private Inner class is created via public Outer class. Outer class private members can be accessed by inner class

Java: inner class

Java: Outer class and inner class example

```java
package CSE1203;

import java.awt.Point;

public class Outer{
private int x;
class Inner{
private int y;
public Inner(int y) {
this.y=y;
}
public void setY(int y) {
this.y=y;
}
public int getY() {
return y;
}

}
public void Display() {
Inner inner=new Inner(0);
inner.setY(12);
System.out.println(inner.getY());
}
}
```

Java: Inner class example

```java
package CSE1203;
public class First {

public static void main(String[] args) {
//outer is Outer class object
Outer outer=new Outer();
outer.Display();
//in is Inner class object
Outer.Inner in = new Outer().new Inner(45);
in.sum();
}
}
```

```java
package CSE1203;

class Outer{
private int x=20;
class Inner{
private int y;
public Inner(int y) {
this.y=y;
}
public void setY(int y) {
this.y=y;
}
public int getY() {
return y;
}
```

```java
public void sum() {
System.out.println("inside sum="+(x+y));
}
}
public void Display() {
Inner inner=new Inner(0);
inner.setY(12);
System.out.println(inner.getY());
}
}
```

THANK YOU