First Simple program in Java

```
1  public class Simple
2  {
3        public  static  void  main  (String args [])
4        {
5              System.out.println("Hello Java");
6        }
7  }
```

To setup Javac path: https://www.Javatpoint.com/how-to-set-path-in-Java

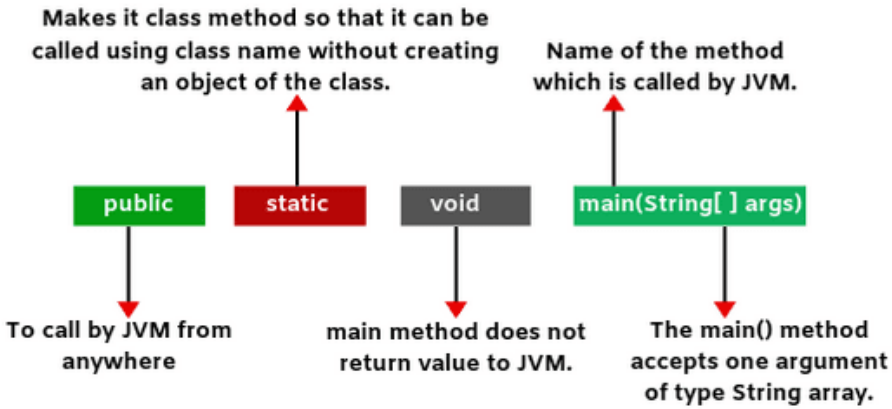
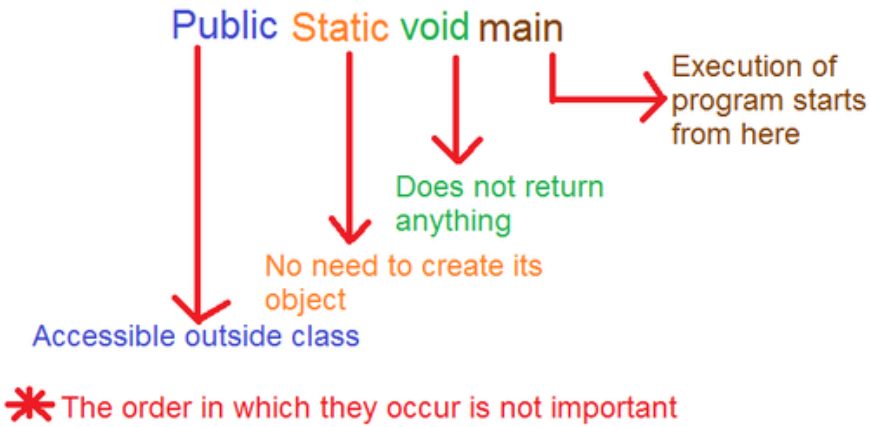


Fig: Java main method

Object Oriented Programming Concepts III



Parameters used in First Java Program:

Let 's see what is the meaning of class, public, static, void, main, String[], System.out.println().

✓ class keyword is used to declare a class in Java.

✓ public keyword is an access modifier which represents visibility. It means it is visible to all.

✓ static is a keyword. If we declare any method as static, it is known as the static method. The core advantage of the static method is that there is no need to create an object to invoke the static method.

Object Oriented Programming Concepts V

✓ The main method is executed by the JVM, so it doesn't require to create an object to invoke the main method. So it saves memory.

✓ void is the return type of the method. It means it doesn't return any value.

✓ main represents the starting point of the program.

✓ String[] args is used for command line argument.

Compare with the arguments of main in C programming language: **int main(int argc, char *argv[])**

✓ System.out.println() is used to print statement. Here, System is a class, out is the object of PrintStream class, println() is the method of PrintStream class.

Object Oriented Programming Concepts VI

- Java file name be the same as the public class name that contains main method. Java file name must be ”Simple.Java”.

```
1  public class Simple
2  {
3        public static void main(String args[])
4        {
5              System.out.println("Hello Java");
6        }
7  }
```

- main method name must be within the public class.

Object Oriented Programming Concepts VII

- One Java file can consist of multiple classes with the restriction that only one of them can be public.

```
1  public class Class1
2  {
3
4  }
5  class Class2
6  {
7
8  }
9  class Class3
10 {
11
12 }
```

Object Oriented Programming Concepts VIII

- Compiler generates separate .class file of all classes after compilation of Java file.

```
1  public class Class1
2  {
3        public static void main(String args[])
4        {
5              System.out.println("Hello Java");
6        }
7  }
8  class Class2
9  {
10
11 }
12 class Class3
13 {
14
15 }
```

Object Oriented Programming Concepts IX

After compilation of Class1.Java, it generates three class files, also called bytecode, Class1.class, Class2.class and Class3.class.

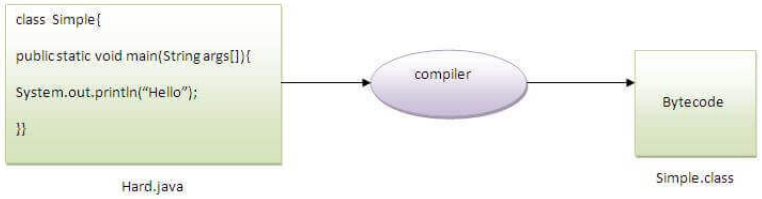
- A class file that contains the main method can be used for execution. In the above example, only Class1.class can be the executable file.

Object Oriented Programming Concepts X

Can you save a Java source file by other name than the class name?

Object Oriented Programming Concepts XI

Yes, if the class is not public. It is explained in the figure given below:

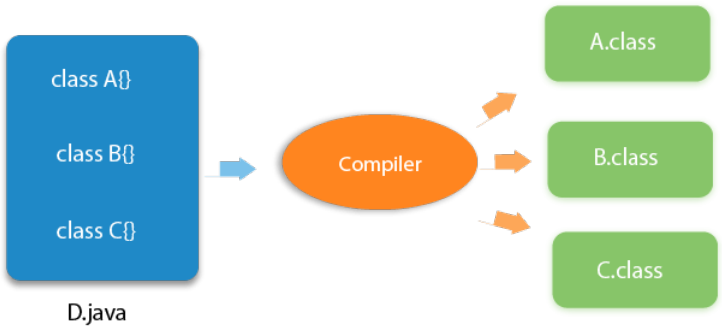


Object Oriented Programming Concepts XII

Can you have multiple classes in a Java source file?

Object Oriented Programming Concepts XIII

Yes, like the figure given below illustrates:



How many ways can we write a Java program:

There are many ways to write a Java program. The modifications that can be done in a Java program are given below:

1. By changing the sequence of the modifiers, method prototype is not changed in Java.
   Let' s see the simple code of the main method.

```
1 static public void main (String args [])
```

2. The subscript notation in Java array can be used after type, before the variable or after the variable.Let 's see the different codes to write the main method.

Object Oriented Programming Concepts XV

```
1 public static void main(String[] args)
2 public static void main(String []args)
3 public static void main(String args[])
```

3. You can provide var-args support to the main method by passing 3 ellipses (dots)
   Let' s see the simple code of using var - args in the main method.We will learn about var - args later in Java New Features chapter.

```
1 public static void main (String ... args)
```

4. Having a semicolon at the end of class is optional in Java.Let 's see the simple code.

Object Oriented Programming Concepts XVI

```
1  class A
2  {
3        static public void main(String... args)
4        {
5              System.out.println("hello Java4");
6        }
7  }
8  ;
```

Valid Java main method signature:

```
1 public static void main(String[] args)
2 public static void main(String []args)
3 public static void main(String args[])
4 public static void main(String... args)
5 static public void main(String[] args)
6 public static final void main(String[] args)
7 final public static void main(String[] args)
8 final strictfp public static void main(String[] args)
```

Object Oriented Programming Concepts XVIII

Invalid Java main method signature:

```
1 public void main(String[] args)
2 static void main(String[] args)
3 public void static main(String[] args)
4 abstract public static void main(String[] args)
```

Java Naming conventions I

✓ Java naming convention is a rule to follow as you decide what to name your identifiers such as class, package, variable, constant, method, etc.

✓ But, it is not forced to follow. So, it is known as convention not rule. These conventions are suggested by several Java communities such as Sun Microsystems and Netscape.

✓ All the classes, interfaces, packages, methods and fields of Java programming language are given according to the Java naming convention. If you fail to follow these conventions, it may generate confusion or erroneous code.

▶ **Advantage of naming conventions in Java**:

Java Naming conventions II

✓ By using standard Java naming conventions, you make your code easier to read for yourself and other programmers. Readability of Java program is very important.

✓ It indicates that less time is spent to figure out what the code does.

The following are the key rules that must be followed by every identifier:

- The name must not contain any white spaces.
- The name should not start with special characters like & (ampersand), $ (dollar), _ (underscore).
- Example: 4count, high-temp, Not/ok are invalid identifiers.

Java Naming conventions III

Let' s see some other rules that should be followed by identifiers.

Class

- It should start with the uppercase letter.
- It should be a noun such as Color, Button, System, Thread, etc.
- Use appropriate words, instead of acronyms.

Example:-

```
1 public class Employee
2 {
3       //code snippet
4 }
```

Interface

- It should start with the uppercase letter.
- It should be an adjective such as Runnable, Remote, ActionListener.
- Use appropriate words, instead of acronyms.

Example:-

```
1 interface Printable
2 {
3       //code snippet
4 }
```

Method

- It should start with lowercase letter.
- It should be a verb such as main(), print(), println().
- If the name contains multiple words, start it with a lowercase letter followed by an uppercase letter such as actionPerformed().

Example:-

```
1 public class Employee
2 {
3       //method void draw()
4       {
5             //code snippet
6       }
7 }
```

Variable

- It should start with a lowercase letter such as id, name.
- It should not start with the special characters like &(ampersand), $(dollar), _(underscore).
- If the name contains multiple words, start it with the lowercase letter followed by an uppercase letter such as firstName, lastName.
- Avoid using one - character variables such as x, y, z.

Example:-

Java Naming conventions VII

```
1  class Employee
2  {
3        //variable
4        int
5        id;
6        //code snippet
7  }
```

Java Naming conventions VIII

Package

- It should be a lowercase letter such as Java, lang.
- If the name contains multiple words, it should be separated by dots (.)such as Java.util, Java.lang.

Example:-

```
1  package com.Javatpoint; //package
2  class Employee
3  {
4        //code snippet
5  }
```

Java Naming conventions IX

Constant

- It should be in uppercase letters such as RED, YELLOW.
- If the name contains multiple words, it should be separated by an underscore (˙)such as MAX˙PRIORITY.
- It may contain digits but not as the first letter.

Example: -

```
1 class Employee
2 {
3       //constant
4       static final int MIN_AGE = 18;
5       //code snippet
6 }
```

Java Naming conventions X

CamelCase in Java naming conventions:

✓ Java follows camel - case syntax for naming the class, interface, method, and variable.

✓ If the name is combined with two words, the second word will start with uppercase letter always such as actionPerformed (), firstName, ActionEvent, ActionListener, etc.

Java Keywords I

► **Separators:** In Java, there are a few characters that are used as separators. The most commonly used separator in Java is the semicolon, and it is used to terminate statements.

Symbol	Name	Purpose
()	Parentheses	Used to contain lists of parameters in method definition and invocation. Also used for defining precedence in expressions, containing expressions in control statements, and surrounding cast types.
{ }	Braces	Used to contain the values of automatically initialized arrays.Also used to define a block of code, for classes, methods, and local scopes.
[]	Brackets	Used to declare array types. Also used when dereferencing array values.
;	Semicolon	Terminates statements.
,	Comma	Separates consecutive identifiers in a variable declaration. Also used to chain statements together inside a for statement

Java Keywords II

The Java Keywords: There are 50 keywords currently defined in the Java language.These keywords, combined with the syntax of the operators and separators, form the foundation of the Java language.

✓ These keywords cannot be used as identifiers.Thus, they cannot be used as names for a variable, class, or method.

✓ The keywords const and goto are reserved but not used.

✓ In addition to the keywords, Java reserves the following: **true, false, and null**. These are values defined by Java. You may not use these words for the names of variables, classes, and so on.

Java Keywords III

abstract	continue	for	new	switch
assert	default	goto	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp	volatile
const	float	native	super	while

Java Keywords IV

1	abstract:	Java abstract keyword is used to declare abstract class Abstract class can provide the implementation of interface It can have abstract and non - abstract methods
2	assert:	Assertion is a statement in Java It can be used to test your assumptions about the program While executing assertion, it is believed to be true If it fails, JVM will throw an error named AssertionError It is mainly used for testing purpose It provides an effective way to detect and correct programming errors
3	boolean:	Java boolean keyword is used to declare a variable as a boolean type It can hold True and False values only

Java Keywords V

4	break:	Java break keyword is used to break loop or switch statement It breaks the current flow of the program at specified condition
5	byte:	Java byte keyword is used to declare a variable that can hold an 8 - bit data values
6	case:	Java case keyword is used to with the switch statements to mark blocks of text
7	catch:	Java catch keyword is used to catch the exceptions generated by try statements It must be used after the try block only
8	char:	Java char keyword is used to declare a variable that can hold unsigned 16 - bit Unicode characters

Java Keywords VI

9	class:	Java class keyword is used to declare a class
10	continue:	Java continue keyword is used to continue the loop It continues the current flow of the program and skips the remaining code at the specified condition
11	default:	Java default keyword is used to specify the default block of code in a switch statement
12	do:	Java do keyword is used in control statement to declare a loop It can iterate a part of the program several times
13	double:	Java double keyword is used to declare a variable that can hold a 64 - bit floating - point numbers
14	else:	Java else keyword is used to indicate the alternative branches in an if statement

Java Keywords VII

15	enum:	Java enum keyword is used to define a fixed set of constants Enum constructors are always private or default
16	extends:	Java extends keyword is used to indicate that a class is derived from another class or interface
17	final:	Java final keyword is used to indicate that a variable holds a constant value It is applied with a variable It is used to restrict the user
18	finally:	Java finally keyword indicates a block of code in a try - catch structure This block is always executed whether exception is handled or not
19	float:	Java float keyword is used to declare a variable that can hold a 32 - bit floating - point number

Java Keywords VIII

20	for:	Java for keyword is used to start a for loop It is used to execute a set of instructions / functions repeatedly when some conditions become true If the number of iteration is fixed, it is recommended to use for loop
21	if:	Java if keyword tests the condition It executes the if block if condition is true
22	implements:	Java implements keyword is used to implement an interface
23	import:	Java import keyword makes classes and interfaces available and accessible to the current source code
24	instanceof:	Java instanceof keyword is used to test whether the object is an instance of the specified class or implements an interface

Java Keywords IX

25	int:	Java int keyword is used to declare a variable that can hold a 32 - bit signed integer
26	interface:	Java interface keyword is used to declare an interface It can have only abstract methods
27	long:	Java long keyword is used to declare a variable that can hold a 64 - bit integer
28	native:	Java native keyword is used to specify that a method is implemented in native code using JNI (Java Native Interface)
29	new:	Java new keyword is used to create new objects
30	null:	Java null keyword is used to indicate that a reference does not refer to anything It removes the garbage value

Java Keywords X

31	package:	Java package keyword is used to declare a Java package that includes the classes
32	private:	Java private keyword is an access modifier It is used to indicate that a method or variable may be accessed only in the class in which it is declared
33	protected:	Java protected keyword is an access modifier It can be accessible within package and outside the package but through inheritance only It can ’t be applied on the class
34	public:	Java public keyword is an access modifier It is used to indicate that an item is accessible anywhere It has the widest scope among all other modifiers
35	return:	Java return keyword is used to return from a method when its execution is complete

Java Keywords XI

36	short:	Java short keyword is used to declare a variable that can hold a 16-bit integer
37	static:	Java static keyword is used to indicate that a variable or method is a class method The static keyword in Java is used for memory management mainly
38	strictfp:	Java strictfp is used to restrict the floating-point calculations to ensure portability
39	super:	Java super keyword is a reference variable that is used to refer parent class object It can be used to invoke immediate parent class method
40	switch:	The Java switch keyword contains a switch statement that executes code based on test value The switch statement tests the equality of a variable against multiple values

Java Keywords XII

41	synchronized:	Java synchronized keyword is used to specify the critical sections or methods in multithreaded code
42	this:	Java this keyword can be used to refer the current object in a method or constructor
43	throw:	The Java throw keyword is used to explicitly throw an exception The throw keyword is mainly used to throw custom exception It is followed by an instance
44	throws:	The Java throws keyword is used to declare an exception Checked exception can be propagated with throws
45	transient:	Java transient keyword is used in serialization If you define any data member as transient, it will not be serialized

Java Keywords XIII

46	try:	Java try keyword is used to start a block of code that will be tested for exceptions The try block must be followed by either catch or finally block
47	void:	Java void keyword is used to specify that a method does not have a return value
48	volatile:	Java volatile keyword is used to indicate that a variable may change asynchronously
49	while:	Java while keyword is used to start a while loop This loop iterates a part of the program several times If the number of iteration is not fixed, it is recommended to use while loop

Java Keywords XIV

Java Data Types

✓ Java Is a Strongly Typed Language

✓ Indeed, part of Java's safety and robustness comes from this fact.

✓ Every variable has a type, every expression has a type, and every type is strictly defined.

✓ All assignments, whether explicit or via parameter passing in method calls, are checked for type compatibility.

✓ There are no automatic coercions or conversions of conflicting types as in some languages.

Data Types, Variables, and Arrays II

✓ The Java compiler checks all expressions and parameters to ensure that the types are compatible.

✓ Any type mismatches are errors that must be corrected before the compiler will finish compiling the class.

Data Types, Variables, and Arrays III

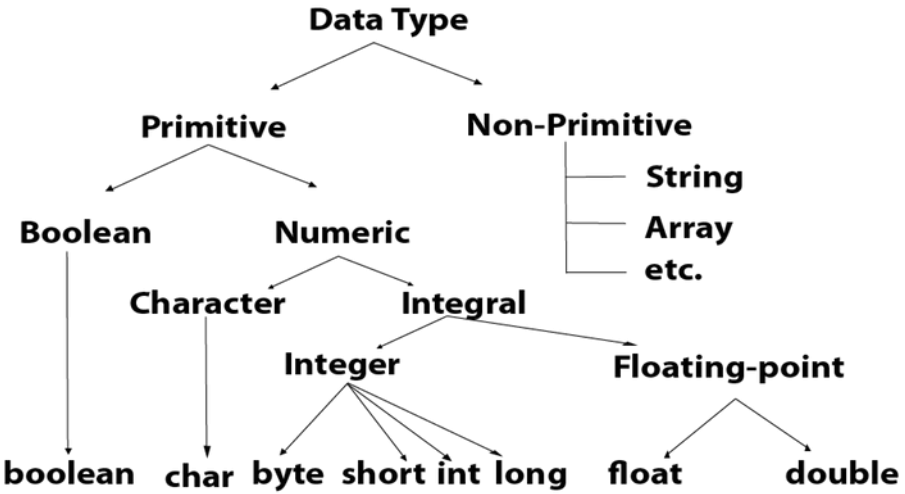
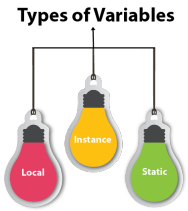


Figure 1: Java Data type

▶ Types of Variables:

There are three types of variables in Java:

1. local variable
2. instance variable
3. static variable



Types of Variables II

1. Local Variable: A variable declared inside the body of the method is called local variable. You can use this variable only within that method and the other methods in the class aren't even aware that the variable exists.
   Can a local variable be defined with " static " keyword?
   If no, then why ????????

Types of Variables III

In Java, a static variable is a class variable (for whole class). So if we have static local variable (a variable with scope limited to function), it violates the purpose of static. Hence compiler does not allow static local variable.

2. Instance Variable: A variable declared inside the class but outside the body of the method, is called instance variable. It is not declared as static.
   - Instance variables are declared in a class, but outside a method, constructor or any block.

Types of Variables IV

- When space is allocated for an object in the heap, a slot for each instance variable value is created.
- Instance variables are created when an object is created with the use of the keyword 'new' and destroyed when the object is destroyed.
- Instance variables 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 at the class level before or after use.
- Access modifiers can be given for instance variables.

Types of Variables V

- The instance variables are 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 (when instance variables are given accessibility), they should be called using the fully qualified name. ObjectReference.VariableName.

Types of Variables VI

It is called instance variable because its value is instance specific and is not shared among instances.

```
1  import java.io.*;
2  public class Employee
3  {
4
5        // this instance variable is visible for any child class.
6        public String name;
7
8        // salary variable is visible in Employee class only.
9        private double salary;
10
11       // The name variable is assigned in the constructor.
12       public Employee (String empName)
13       {
14             name = empName;
15       }
16
17       // The salary variable is assigned a value.
18       public void setSalary(double empSal)
19       {
```

Types of Variables VII

```
20              salary = empSal;
21        }
22
23        // This method prints the employee details.
24        public void printEmp()
25        {
26              System.out.println("name : " + name );
27              System.out.println("salary :" + salary);
28        }
29
30        public static void main(String args[])
31        {
32              Employee empOne = new Employee("Ransika");
33              empOne.setSalary(1000);
34              empOne.printEmp();
35        }
36  }
```

Types of Variables VIII

3. Static variable: A variable which is declared as static is called static variable.
   - ✓ It cannot be local.
   - ✓ You can create a single copy of static variable and share among all the instances of the class.
   - ✓ Memory allocation for static variable happens only once when the class is loaded in the memory.

Types of Variables IX

```
1  public class A
2  {
3        int data=50; //instance variable
4        static int m=100; //static variable
5        void method()
6        {
7              int n=90; //local variable
8        }
9  }
10 //end of class
```

static is a non-access modifier in Java which is applicable for the following:

A. blocks
B. variables

Types of Variables X

C. methods

D. nested classes

A. Blocks:

✓ To create a static member (block,variable,method,nested class), precede its declaration with the keyword static.

✓ When a member is declared static, it can be accessed before any objects of its class are created, and without reference to any object.

Types of Variables XI

Static blocks in Java: Java supports a special block, called static block (also called static clause) which can be used for static initializations of a class.

```
1  public class Test
2  {
3        static int i;
4        int j;
5
6        // start of static block
7        static
8        {
9              i = 10;
10             System.out.println("static block called ");
11       }
12       // end of static block
13 }
14 class Main
15 {
16       public static void main(String args[])
17       {
```

Types of Variables XII

```
18              // Although we don't have an object of Test, static block is
19              // called because i is being accessed in following statement.
20              System.out.println(Test.i);
21      }
22  }
```

✓ Also, static blocks are executed before constructors.

```
1   class Test
2   {
3       static int i;
4       int j;
5       static
6       {
7           i = 10;
8           System.out.println("static block called ");
9       }
10      Test()
11      {
12          System.out.println("Constructor called");
13      }
14  }
15  public class Main
```

Types of Variables XIII

```
{
      public static void main(String args[])
      {
            // Although we have two objects, static block is executed only once.
            Test t1 = new Test();
            Test t2 = new Test();
      }
}
Output:
static block called
Constructor called
Constructor called
```

Types of Variables XIV

```java
// Java program to demonstrate use of static blocks
public class Test
{
	// static variable
	static int a = 10;
	static int b;
	// static block
	static
	{
		System.out.println("Static block initialized.");
		b = a * 4;
	}
	public static void main(String[] args)
	{
		System.out.println("from main");
		System.out.println("Value of a : "+a);
		System.out.println("Value of b : "+b);
	}
}
```

Types of Variables XV

- Static Variables: When a variable is declared as static, then a single copy of variable is created and shared among all objects at class level. Static variables are, essentially, global variables. All instances of the class share the same static variable.
  Important points for static variables :-
  - We can create static variables at class-level only.
  - static block and static variables are executed in order they are present in a program.

Types of Variables XVI

```
1  // java program to demonstrate execution of static blocks and variables
2  public class Test
3  {
4        // static variable
5        static int a = m1();
6        // static block
7        static
8        {
9              System.out.println("Inside static block");
10       }
11       // static method
12       static int m1()
13       {
14             System.out.println("from m1");
15             return 20;
16       }
17       public static void main(String[] args)
18       {
19             System.out.println("Value of a : "+a);
20             System.out.println("from main");
21       }
22 }
```

Types of Variables XVII

- Static methods: When a method is declared with static keyword, it is known as static method.
  - ✓ The most common example of a static method is main( ) method.
  - ✓ Any static member can be accessed before any objects of its class are created, and without reference to any object.
  - ✓ Methods declared as static have several restrictions:
    - They can only directly call other static methods.
    - They can only directly access static data.
    - They cannot refer to this or super in any way.

Types of Variables XVIII

✓ For example, in below java program, we are accessing static method m1() without creating any object of Test class.

```java
// Java program to demonstrate that a static member
// can be accessed before instantiating a class
public class Test
{
      // static method
      static void m1()
      {
            System.out.println("from m1");
      }
      public static void main(String[] args)
      {
            // calling m1 without creating
            // any object of class Test
            m1();
      }
}
```

Types of Variables XIX

```java
// java program to demonstrate restriction on static methods
public class Test
{
	// static variable
	static int a = 10;
	// instance variable
	int b = 20;
	// static method
	static void m1()
	{
		a = 20;
		System.out.println("from m1");

		// Cannot make a static reference to the non-static field b
		b = 10; // compilation error

		// Cannot make a static reference to the non-static method m2() from the type Test
		m2(); // compilation error

		// Cannot use super in a static context
		System.out.println(super.a); // compiler error
	}
```

Types of Variables XX

```
23        // instance method
24        void m2()
25        {
26              System.out.println("from m2");
27        }
28        public static void main(String[] args)
29        {
30              // main method
31        }
32  }
```

Types of Variables XXI

When to use static variables and methods?

✓ Use the static variable for the property that is common to all objects.

✓ For example, in class Student, all students shares the same college name. Use static methods for changing static variables.

```
1  // A java program to demonstrate use of static keyword with methods and variables Student class
2  class Student
3  {
4        String name;
5        int rollNo;
6        // static variable
7        static String cllgName;
8        // static counter to set unique roll no
9        static int counter = 0;
10       public Student(String name)
11       {
12             this.name = name;
13             this.rollNo = setRollNo();
14       }
```

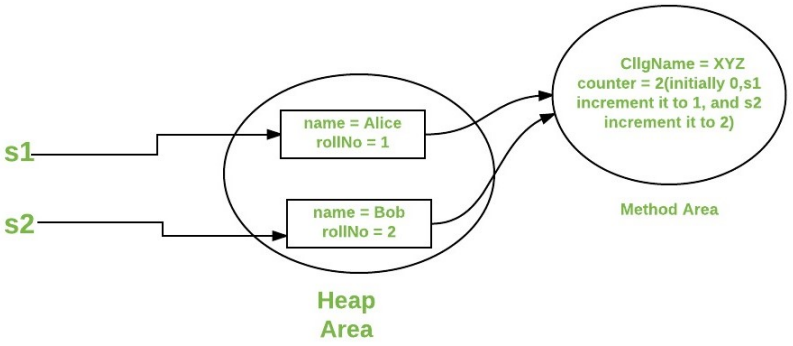
Types of Variables XXII

```
15      //getting unique rollNo through static variable(counter)
16      static int setRollNo()
17      {
18            counter++;
19            return counter;
20      }
21      // static method
22      static void setCllg(String name)
23      {
24            cllgName = name ;
25      }
26      // instance method
27      void getStudentInfo()
28      {
29            System.out.println("name : " + this.name);
30            System.out.println("rollNo : " + this.rollNo);
31            // accessing static variable
32            System.out.println("cllgName : " + cllgName);
33      }
34  }
35  //Driver class
36  public class StaticDemo
37  {
```

Types of Variables XXIII

```
38        public static void main(String[] args)
39        {
40              // calling static method without instantiating Student class
41              Student.setCllg("XYZ");
42              Student s1 = new Student("Alice");
43              Student s2 = new Student("Bob");
44              s1.getStudentInfo();
45              s2.getStudentInfo();
46        }
47  }
48  Output:
49
50  name : Alice
51  rollNo : 1
52  cllgName : XYZ
53  name : Bob
54  rollNo : 2
55  cllgName : XYZ
```

Types of Variables XXIV



Types of Variables XXV

1. Static nested classes : We can not declare top-level class with a static modifier, but can declare nested classes as static. Such type of classes are called Nested static classes.

```
1  public class TestOuter1
2  {
3        static int data=30;
4        static class Inner
5        {
6              void msg()
7              {
8                    System.out.println("data is "+data);
9              }
10       }
11       public static void main(String args[])
12       {
13             TestOuter1.Inner obj=new TestOuter1.Inner();
14             obj.msg();
15       }
16 }
```

Types of Variables XXVI

```
Output:

data is 30
```

Types of Variables XXVII