1 JAVA ORIENTATION

Section- 1

- Any problem, any logic, for that matter, anything can or rather has to be put inside a CLASS.in JAVA.

- So fill up tour code in a class and save it as class name.java(*preferably*)
  Example:

```
class ABC
    {
        int a;
        String s;
        void a()
            {
                System.out.println("amethod");
            }
    }
```

Class name for the above program is:**ABC.java** (*preferable but not compulsary*)

Note:But if you make a class public, you must name that file with that class name.

Example:

```
public class ABC
    {
        int a;
        String s;
        void a()
            {
                System.out.println("amethod");
            }
    }
```

Class name is : **ABC.java** (*MANDATORY*)
CDE.java (*COMPILATION ERROR*)

- A .java file can contain more than one class but only public class. **Example:**

```
class ABC
    {
        int i;
        String s;
        void a()
            {
                System.out.println("HI");
            }
    }
class B
    {
        char c;
        void b()
            {
                System.out.println("HELLO");
            }
```

```
}
```

Class name for the above program can be : **A.java**
(*or*)
B.java
(*or*)
some random name.java

But, The following program will fail.

Example:

```
public class ABC
    {
        int i;
        String s;
        void a()
            {
                System.out.println("HI");
            }
    }                   public class B
    {
        char c;
        void b()
            {
                System.out.println("HELLO");
            }
    }
```

Class name for the above program: **A.java**
(*or*)
B.java

THESE TWO CLASS NAMES ARE WRONG.

Reason : A .java should not contain more than one public class.So, the correct version is the following:

Example:

```
public class A
    {
        int i;
        String s;
        void a()
            {
                System.out.println("HI");
            }
    }
class B
    {
        char c;
        void b()
            {
                System.out.println("HELLO");
            }
    }
```

Class name for the above program: **A.java** (*COMPULSORY*)

EXAMPLES OF OTHER CLASSES

eg:1

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

Class name is **Test.java**

eg:2

```
class Student
{
    int rollno;
    String name;
    String degree;
    void read()
    {
        System.out.println("Reading");
    }
    void write()
    {
        System.out.println("writing");
    }
}
```

Section-2

Any .java has to pass through two phases:

1.Compilation Phase
2.RunTme Phase

1.Compilation Phase:

Any java program is compiled using "javac filename.java"

Example:1

```
class ABC
{
    int i;
    void a()
    {
        System.out.println("HI");
    }
}
```

ABC.java(*classname*)$\longrightarrow$javac ABC.java(*compiling*)$\longrightarrow$ABC.class

After Compilation classname.class will be produced.

Example:2

```
public class ABC
{
```

```
        int i;
        void b()
            {
                System.out.println("HI");
            }
    }
class B
    {
        B()
            {
                System.out.println("B");
            }
    }
class C
    {
        void d()
            {
                System.out.println("d");
            }
    }
```

Class name for the above program: **ABC.java** and after compiling the above program using javac ABC.java three .classes will be produced i.e. **ABC.class, B.class, c.class**.

What happens in Compilation error

Compliler will check for the following:

1.Syntax Checking

```
//A.java
class A
    {
        int i⟶(compilation falis because no semicolon;)
int j;
        void a();⟶(Complilation falis because semicolon should not be placed during defination of method)
        {
            System.out.println("d");
        }
    }
```

2.Wrong Assignments

```
//A.java
eg:1    class A
        {
            public static void main(String[]args)
            {
                float f = 19.1;⟶(Compilation falis because assigning double(19.1) to float f will cause of precision)
            }
        }
```

eg:2 More common case is with "reference" variables.

```
//Cat.java
class Cat
    {
        int size;
        int height;
        void talk()
            {
                System.out.println("meow");
            }
    }
// Dog.java
class Dog
            {
        int size;
        int height;
        void talk()
            {
                System.out.println("bow");
            }
    }
//Test.java
    {
        public static void main(String[]args)
            {
                Cat c= new Cat();
                Dog d= new Dog();
                d=c;⟶(Compilation will fail because assign "Cat" variable to "Dog" will not be allowed)
            }
    }
```

3.Missing Compulsory Statements

If you have to write some statements for sure, but if you miss them, compiler will catch them:

Example:1 A.java

```
class A
    {
        int kk()
            {
                System.out.println("i");⟶(Compilation will falis because "return" statement is missing)                }
    }
```

Example:2 A.java

```
class A
    {
        int decision(int a)
            {
                if(a¿10)
```

```
                    {
                        return a;——>(Compilation falis because return is placed in "Optional Block")
                    }
                else if(a¿10)
                    {
                        return a+2;——>(Compilation falis because return is placed in "Optional Block")
                    }
                }
        }
```

Example:3 A.java

```
class A
    {
        int decision(int a)
        {
            if(a¿10)
                {
                    return a;——>(Compilation falis because return is placed in "Optional Block")
                }
            else if(a¿10)
                {
                    return a+2;——>(Compilation falis because return is placed in "Optional Block")
                }
        }
    }
```

Example:4 A.java

```
class A
    {
        int decision()
        {
            for(int i = 0; i ¡ 10; i++)
                {
                    return 1;——>(Compilation falis because return is placed in "Optional Block")
                }
        }
    }
```

4.Putting Unreachable Statements

Anything written after "return statement" become unreachable, Hence throws an error.

Example:1 A.java

```
class A
    {
        int kk()
        {
```

```
                    System.out.println("kk");
                        return 10
                        System.out.println("kk");⟶(Compilation
fails because return is placed in "Optional Block")
                }
        }
```

5.Violation Of rules There are many kind of java rules. If any of the rules are violated, compiler will shout saying, we have violated.

Example:1 Accesing a private variable

A.java

```
class A
    {
        public static void main(String[]args)
            {
                A a = new A();
                a.i = 10⟶(Accessing a private variable
of Other Class)
            }
    }
```

Example:2 Accesing a state variable(without creating an object) From Static Context.

Test.java

```
class Test
    {
        int i; // state variable
        public static void main(String[]args)
            {
                i = 0;⟶(This is wrong because 'i' is a
non-static while main is static)
            }
    }
```

Example:3

Test.java

```
class Test
    {
        int i;
        static int j;
        public static void main(String[]args)
            {
                j = 0;⟶(This is correct because 'j' is a
static while main is static)
            }
    }
```

Example:4

// A.java

```
class A
    {
        int i;
        static int j;
    }
// Test.java
class Test
    {
        public static void main(String[]args)
            {
                System.out.println("A.i");——→(This will fail because 'i' is a static variable, have to create an object to access it)
            }
    }
```

6.Checked Exceptions:(later)

Note:Like this, there are many type of compilation errors, So before one starts looking for output, check whether program compiles or not.

2.Runtime Environment:

1) After "Successful Compilation" , A class with "MAIN" method will execute.

Example1: //Test.java

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

javac Test.java(Compiling)
↓
Test.class(Successful Compilation)
↓
java Test("java" is the command to run a java program)

Example2: //A.java

```
class A
    {
        int i;
        void a()
            {
                System.out.println("Hai");
            }
    }
```

javac A.java→(Success)

java A→(This will fail because no main method)

2) As soon as you see successful compilation and a class with "main" method, start drawing STACK and HEAP structure to analyse. STACK and HEAP concept is discussed in chapter 4.

3) The "ARGUMENTS" of a main method are special and are called "Command Line Arguments" or "CLA".

Example1: //Test.java

```
class Test
    {
        public static void main(String[]args)
            {
              System.out.println(args[0]);
            }
    }
```

javac Test.java(Compiling)

java Test Example

Output: Example

Example2: Another common example of CLA is reading integer and add them or some operations. But since you are inputting them as strings, you have to convert string to integer. We use Integer.parseInt() method to achieve that.

//Test.java

```
class Test
    {
        public static void main(String[]args)
            {
              int a = Integer.parseInt(args[0]);
              int b = Integer.parseInt(args[1]);
              System.out.println(a+b);
            }
    }
```

javac Test.java(Compiling)

java Test 23 114 → Output:137

java Test 1 4 → Output:5

But if you say just

java Test → Exception

java Test 1 4 → ArrayOut of Bounds Exception

4) One more famous RunTime error is "stack overflow error".

Example: //A.java

```
class A
    {
        void a()
            {
              System.out.println("Hai");
              a();
            }
    }
//Test.java
class Test
    {
        public static void main(String[]args)
            {
              A a = new A();
              a.a();
            }
    }
```

5) One more famous RunTime exception is Null Pointer Exception.

```
//Test.java
class Test
    {
        public static void main(String[]args)
            {
              A a = new;
              a.a();
            }
    }
```

In this code without creating object, accessing state methods/ variables with "null" reference.

Note:In all the above casses Compilation Succeeds.