Week 1

24 June 2023 08:02 PM

24/06/2023

A.java -> A.class (after compilation)

Constructor- used to initialize instance variables.

A.java

```
public class A{

}
```

A.class

```
public class A{
        public A(){

        }
}
```

When we don't write constructor in the class, then in the compiled class, default constructor is written by Java

----

A.java

```
public class A{
        int x= 10;
}
```

A.class

```
public class A{
        int x;
        public A(){
                x= 10;
        }
}
```

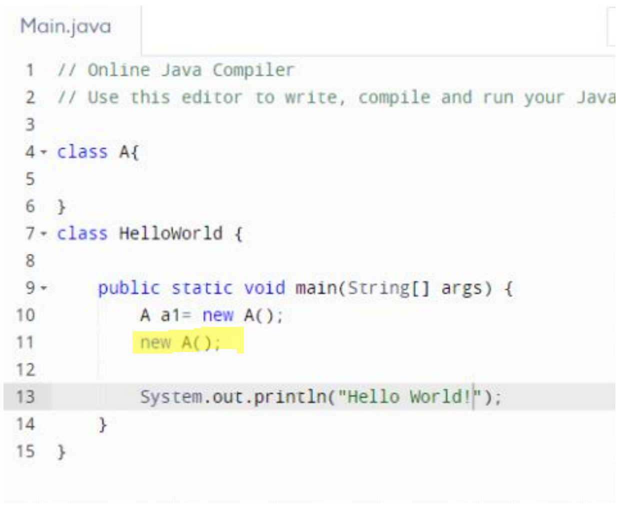
If we've already written the constructor in java class, then constructor is not put in the byte code.

We can write multiple constructors in single class.

Constructor gets called automatically whenever new object of class will get created.

A a1= **new A();** // because of new A() constructor will be called

Create a file- A.java
To compile this file- **javac fileName.java**
To run the class inside this file- java className


Main.java

```
// Online Java Compiler
// Use this editor to write, compile and run your Java

class A{

}
class HelloWorld {

    public static void main(String[] args) {
        A a1= new A();
        new A();

        System.out.println("Hello World!");
    }
}
```

Inside the method, I've written new A();
But this new location cannot be used, as there is not reference to use it

Eclipse
Debugging

- Set breakpoint
- Right click in the code-> Debug As -> java application

this- current object

float salary = 200;
//error- float salary= 200.23;
float salary= 200.23**f** ;

java.lang package is by default imported. So, classes like String, System are not needed to be imported.

Scanner class was added in Java 1.5

public **final** class Scanner

Scanner class is final class. So that its child class cannot be created. It is used get input value during running of program. It can get input from keyboard and other resources like I/O files.

import java.util.Scanner;

Before running of program- Input can be taken from String[] args
During running of program- Input can be taken from Scanner.

```
public class A{
        int x= 10;  // this is a global variable. Instance variable.

        public static void main(){
                A a1= new A();
                System.out.println(a1.x);
        }
}
```

Modulus-
19 % 25 Answer:- 19

Types of global variable- 1. static 2. instance(non-static)

char has no default value.
char- 2 bytes
boolean- 1 bit
short- 2 bytes

Primitive data types- int, short, byte, long, float, double, char, boolean

Instance variables get the default value.
The variables inside the method do not get the default value.
String[] args- any name can be written instead of args

A.java
```
class B{
   public static void main(){
        System.out.println("Inside class B");
  }
}
```

javac A.java (javac fileName.java)
java B (java classToBeCompiled)
I want to compile class B inside A.java file

If there is going to be any public class in java file, then its name and java file's name should be same.
Otherwise in the above example, there is no public class. So, there are different names.

class body- inside { } of class
method body- inside { } of method

25/06/2023

```
public class A{
        int a, b;

        public void print(int a, int b){
        this.a= 10; // for global variable
        this.b= 20;  // for global variable

        a= 30; // local variable
        a= 40; // local variable
        System.out.println(a+b); //addition of local variables
        }
}
```

** My observations-

1. **Inside static method**-
   - to call non-static method in same class-> Create object and call from object
   - to call non-static variable(it'll be a global variable in class) in same class-> Create object and call from object
   - to call static method in same class-> call directly staticMethodName();

2. **Inside non-static method**-
   - to call static method or to use static variable in same class-> call directly staticMethodName();
   - to call non-static method in same class-> call directly Non-staticMethodName();
   - to call non-static variable in same class-> call directly Non-staticMethodName();

Inheritance

```
public class A{
        int x= 30;
        public void m1(){
                System.out.println("m1--A");
        }
}
```

```
class B extends A{

public void m2(){
  System.out.println("m2--B");
  m1();  // because B extends A
// this is like calling non-static method m1() inside non-static method
m2.
System.out.println(x); //30
}

public static void main(String[] args){
  B b= new B();
  b. m2();
  b.m1();
  System.out.println(b.x);
}
}
```

Parent reference and child object

class A
class B extends class A

A a= new B(); Constructor class

Reference class

For compilation-

- In A a= new B();
  For code to get compiled, only the methods and variables in class A and the parent class of class A, can be called using **a** (reference variable of class A)

For running-

- **Method running rule- After the code is compiled, method running starts from constructor class.**
  If that method is present in constructor class, then it is executed. Otherwise, the method in immediate parent class will be executed. If not present, then the parent class of parent class and so on…

- **Variable running rule- Variable in reference class or in the parent class of reference class can be accessed.**

```
class A{
 int a= 10;

public void m1(){
        System.out.println("m1--A");
}

}
```

```
class B extends A{
int a= 50;
public void m4{
        System.out.println("m4--B");
}

}
```

```
class C extends B{
int a= 90;
public void m4{
        System.out.println("m4--C");
}

public void m5{
        System.out.println("m5--C");
}

}
```

```
B b= new C();
for code get compiled I'll write-
b.m4(); //output-  m4--C  (method-running rule- call from constructor class; if not then its parent)
b.m1();  // output-  m1--A
System.out.println(b.a); // output-  50  (variable running rule- from reference class; if not then its parent)
//error- b.m5();
```

Return type rules in case of inheritance
Co-variant return type

If the method's return type is class type, then we can return same class's reference or their child class's reference.
But we cannot return parent class's reference.

class A
class B extends A
class C extends B

```
public class Test{
        public B m1(){
           A a= new A();
           B b= new B();
           C c= new C();
        // error- return a;
        // possible- return b;
        return c;

        }
```

APPROVED

}

Best practices-
Initialize the global variables in a class(Student class)- Make a method in different class which will create an object of Student class and set values to the global variables in Student class.

```
class A
class B extends A

public A someMethod(){
        A  a= new A();
        B b= new B();

        return a;
        // possible- return b;
}
```

To call this method-

```
A a= new A();
A a1= a.someMethod();
// Here a.someMethod() cannot be stored in class B's object
```