**MOCK Done By Azam Amir Reza (**[**azam20104u@gmail.com**](mailto:azam20104u@gmail.com)**)**

**11-Sep-2015**

1. **What is a class and object?**

Ans: Any entity digit with known class, state and behavior is known as object, the state represent the character of the object whereas the behavior represents functionality of the objects

* A class is design of objects which defines the properties of the objects
* A class is object which defines states and behavior of an object.
* States of the objects are defined by data members whereas the behaviors of the objects are define by non-static function member of the class.

**Class -** A class can be defined as a template/blue print that describes the behaviors/states that object of its type support. A class is a blue print from which individual objects are created.

**Object -** Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors -wagging, barking, and eating. An object is an instance of a class.

Ex, of create objects:

class Circle{

double rad; final static double pi=3.14;

void area() { double a1=pi\*rad\*rad; System.out.println("area is: "+a1); }

void circum() { double c1=2\*pi\*pi; System.out.println("circum is: "+c1); } }

class MC7 { public static void main(String[] args) {

Circle cir1=new Circle(); //create cir1 as an object cir1.rad=2.1; cir1.area(); cir1.circum();

Circle cir2=new Circle(); //create cir2 as an object cir2.rad=3.1; cir2.area(); cir2.circum();

}

}

1. **What is the difference between static and non-static members?**

Ans.

* The static members are declared using static keyword and are also known as class members. The non-static member declared without static keyword it’s also known as instance members of the class.
* Static data members of a class can be rename by any class

The members of a class can be access from any class based on the access specifies

* The static members of a class are refer by using class name

And the non-static data member can be refer using instance name

1. **When to be going for static member and non-static member?**

**Ans.** Static makes part of a Type; others make part of an instance of that type. If you want have some shared state (say) between different instances of the same type, use static. If you want that every instance have its own value, independent from others, use instance fields. you should explain more yourself and your way like your university name is same but the students name and their properties are different. So university is a static and student properties are non-static

1. **What is reference variable and why we need it?**

**Ans.**

* Any references the instance or of the class can be created by using new operator and constructor of the class.
* The new operator is used to instance of class whereas the constructors are used to initialize the instance.
* Whenever we create instance of the class copy of non-static member are loaded in the memory, to refer each instance we should create reference variable or object references.
* Reference variables are special variable which is used to refer an instance of the classes.
* The reference variable should be declared by using class name.
* For a reference variable we can assign either null value or an instance of class.
* An instance can be referred by any no of reference variables. In such case modifying the state of the instance from 1 reference variable will reflect in another reference variable also
* The instance created will always have non-static data member and non-static function member
* The modifying the states of one statements will not update the entire instance.

1. **Does java support pass by reference explain?**

**Ans. *Java does not support pass by reference since java language is not be using pointer concept ,however we can pass the object address to reference variable not true pointers.***

* If the method argument is a class type then while invoking such method we have to pass the instance of the class mentioned in the method argument.
* The instance can be passed directly or we can pass the reference of instance also
* If we passing the reference of instance then any modification to that instance will reflect in the other references

Ex, class Sample1{

int i=34; double j=4.5; void disp() { System.out.println("Runnigg disp of Sample1 "); } } class Demo1 { void test(Sample1 arg1){ //method argument is class type System.out.println("running test() of Demo1 class");

System.out.println("i value: "+arg1.i); //will print I value: 34 System.out.println("j value: "+arg1.j); //will print j value: 4.5 obj1.disp();

arg1.i=87;//reassign arg1.j=7.6;//reassign } } class MainClass1 { public static void main(String[] args) {

Demo1 d1=new Demo1(); Sample1 ref1=new Sample1();//create a reference of Sample1

d1.test(ref1);//value of ref1 is copied to obj1 and passing a reference of Sample1 Class System.out.println("i value: "+ref1.i); //will print I value: 87 System.out.println("j value: "+ref1.j); //will print j value: 7.6

System.out.println("program ended!"); } }

Heap Area

test()

arg1[address]

main()

ref1[address]

Copied

***\*\* Java does not support pass by reference since java language is not be using pointer concept ,however we can pass the object address to reference variable not true pointers.***

1. **What is constructor, why we need it?**

**Ans.**

* Constructors are special member of the class which is used to initialize the data members of the class.
* The constructors are executed whenever the instances of class are created.
* Every class must have constructor in order to create an instance of the class.
* The constructor can be created either by compiler or by user.
* The constructor define by compiler is known as default constructor
* Compiler defines a constructor if the class does not have any user define constructor. If in case the class is having user define constructor then the compiler not define default constructor.
* The constructor define by programmer is known as user define constructor .the user can define either 0 or with argument constructor.
* The constructor defines with arguments constructor are known as parameter constructor.
* Whenever as object created using parameter constructor we need to pass value to the constructor
* The cons cannot be declare as static

1. **What is constructor overloading, why it is required?**

**Ans.**

In a class defining multiple constructor with deferent parameters is known as constructor overloading

* When defining overloading constructor the constructor should deferent in terms of parameter type or parameter length.
* While defining a constructor the constructor name should be same as class name, and constructor should not have return type.
* Constructor overloading help us to create an instance of class to the different initialization.

Ex,

class S4{ int k; double d;

//constructor overloading S4() { System.out.println("running no arg cons"); }

S4(int arg1) { System.out.println("running int arg cons"); k=arg1; }

S4(double arg1) { System.out.println("running double arg cons"); d=arg1; }

S4(int arg1,double arg2) { System.out.println("running int, double arg cons"); k=arg1; d=arg2; }

void test(){ System.out.println("k value: "+k); System.out.println("d value: "+d); } }

class MainClass5 {

public static void main(String[] args) {

S4 obj1=new S4(); obj1.test();

S4 obj2=new S4(23); obj2.test();

S4 obj3=new S4(7.5); obj3.test();

S4 obj4=new S4(45,95.4); obj4.test(); System.out.println("Program Ended"); } }

o/p:

running no arg cons

k value: 0

d value: 0.0

running int arg cons

k value: 23

d value: 0.0

running double arg cons

k value: 0

d value: 7.5

running int, double arg cons

k value: 45

d value: 95.4

Program Ended

1. **What is the used of “this” keyword and how it is different from “super” keyword?**

**Ans.**

* “This” keyword can be used for identify the local variable or global variable of a class body. Such that if in a class body there have a constructor and have a local variable with the name id and the class body there have a global variable with the name id at that time we can use the “this” keyword to identify the global variable.
* Java provides the special keywords by “super” which is used to refer super class properties in subclass calling or from subclass.
* The super keywords should be used either in non-static method context or constructor context.
* It can’t be used in static context.

***Example of using “this” keyword****,* class S1{ int id=123; //id globall variable S1(int id){ // inside the constructor id is local variable this.id=id; //reassign of global variable int id=123 to 456 //identify global variable to the local variable using “this” keyword . } }

class MainClass1 { public static void main(String[] args) {

S1 obj1=new S1(456);//passing the argument to S1(int id ) local variable

System.out.println("id value: "+obj1.id); } }

o/p: id value: 456

***Example of using “super” keyword****,* class D1 { int k=34;

} class S1 extends D1{

int k=56; void disp(){ System.out.println("k value of S1 : "+this.k);//call S1 global variable k=56

System.out.println("k value of D1 : "+super.k);//call super class variable k=34 } }

o/p: k value of S1 :56 k value of D1 : 34

1. **Does constructor return any value? Explain.**

Answer: - yes, that is current class instance (You cannot use return type yet it returns a value).

Inside constructor we can write "return" keyword it will not show any error and if you write "return any-value"   
it will show error.   
  
Constructor return object, and cannot return value and has not return type as well.

Class Test {  
Test () {  
return;  
}  
}   
  
In that case, compiler will not show any error.

1. **What is constructor chaining, when it happens and how?**

**Ans.** Constructor chaining is a phenomenon where subclass constructor makes a call the super class constructor; the super class constructor makes a call to its super class constructor. Constructor chaining can be done either implicitly or explicitly. Constructors are chained and they are called in a particular order

***Exam of Explicitly constructor chaining:***

class D1{ D1(int arg1) { System.out.println("runing const of D1 "); System.out.println("arg1 value: "+arg1); } }

class S1 extends D1{ S1() { super(25);//calling to the super class constructor D1(int arg1) System.out.println("running const of S1"); } }

class MC5 { public static void main(String[] args) {

S1 ref1=new S1(); } }

o/p:

runing const of D1

arg1 value: 25

running const of S1

*If you want implicit then you should declare super class constructor as a zero argument constructor.*

1. **Explain “super ()” statements?**

* “super ()” statements used to make a call to the super class constructor from sub class constructor.
* uper ()” statements should be used only inside the constructor body.
* “super ()” statements should be the first statements of constructor body; it can’t be used anywhere else.
* Multiple super statements is not allowed in the constructor.

Example of “super()” statements see above constructor chaining

1. **Difference between constructor and methods?**

**Answer:**

Main difference between Constructor and Method is that, you need to call method explicitly but constructor is called implicitly by Java programming language during object instantiation. It doesn't mean you cannot call Constructor; if you have [overloaded constructor](http://javarevisited.blogspot.sg/2012/01/what-is-constructor-overloading-in-java.html) than you can call them using [this keyword](http://javarevisited.blogspot.sg/2012/01/this-keyword-java-example-tutorial.html) as this() constructor, you can even call super class constructor using super() keyword, in-fact that is done automatically by Java compiler if no explicitly constructor is called and that is known as **constructor chaining in Java**.

1. First difference between method vs constructor in Java is that **name of constructor** must be same with name of theClass but there is no such requirement for method in Java. methods can have any arbitrary name in Java.
2. Second difference between method and constructor in Java is that **constructor doesn't have any return type** but method has return type and return something unless its void.
3. Third difference between constructor and method in Java is that Constructors are chained and they are called in a particular order, there is no such facility for methods.
4. Unlike method, constructor, yet declared in class doesn't considered as [member of Class](http://javarevisited.blogspot.sg/2011/10/class-in-java-programming-general.html). Constructors are not inherited by child classes but methods are inherited by child classes until they are made private. on which case they are only visible in class on which they are declared. Similarly private constructor means you can not create object of that class from outside, this is one of the technique used to implement [Singleton pattern in Java](http://javarevisited.blogspot.gr/2012/07/why-enum-singleton-are-better-in-java.html).
5. Another difference between method and constructor in Java is that special keyword this and super is used to call constructor explicitly. no such thing for method, they have there own name which can be used to call them.

That's all on difference between method and constructor in Java. You can compare method vs Constructor on different points but main thing is that they are used for object initialization, while method is used to perform a small unit of task.

1. **Different between static and non-static blocks, in details?**

Answer:

class A{

static int i=10;  
//**static block**

static {  
System.out.println("static block");  
}

//**non-static block**{  
System.out.println("not static block");  
}  
}

*A static block is a section of code that gets executes when the class gets loaded.*

E.g.

class MC1 { public static void main(String[] args) { System.out.println("i value: "+A.i); } }

In the main method I just loaded class A with some static int variable .so that my o/p is

Static block I value: 10.

Here not executed the non-static blocks.

*A non-static block is executed when the class is instantiated.*

E.g.

class MC1 { public static void main(String[] args) { A obj1=new A();// instance of class A } }

Here I just create an instance of class A , so that I can executed the non-static block as well as static block because I loaded the class A also. That’s why O/p is

Static block

Non-static block

1. **What is static object and non-static object explain (related to constructor)?**

Answer,

**Non-Static Object:**

It is a reference variable of a class which we can create in another class body. To call that reference variable first we should create an instance of the current class through that instance we can call the reference variable or non-static object.

class Demo1 { int k=12; void test() { System.out.println("running test()"); } } class Sample1 { boolean b=true; Demo1 obj1=new Demo1();//non-st ref variable of type Demo1 void disp() { System.out.println("running disp()");

**Sample 1 HEAP AREA**

Ref1 add **Demo1**

Obj1 add

}

}

----------- --------------------

class MainClass1 { public static void main(String[] args) {

Sample1 ref1=new Sample1(); System.out.println("b value: "+ref1.b); ref1.disp();

System.out.println("k value: "+ref1.obj1.k); ref1.obj1.test(); } }

o/p:

b value: true running disp()

k value: 12 running test()

**Static Object:**

It is also a reference variable of a class which we can create in another class body with “static” keyword. To call that static reference variable, first we should create an instance of the current class and then we can call the reference variable with its class name and reference variable name.

class Demo1 { int k=12; void test() { System.out.println("running test()"); } } class Sample1 { boolean b=true; static Demo1 obj1=new Demo1();//static ref variable of type Demo1 void disp() { System.out.println("running disp()"); } }

--------- ------------

class MainClass1 { public static void main(String[] args) { Sample1 ref1=new Sample1(); System.out.println("b value: "+ref1.b); ref1.disp();

**Sample 1 HEAP AREA**

ref1 **Demo1**

obj1

System.out.println("k value: "+Sample1.obj1.k); Sample1.obj1.test();

**Static Pool Area**

Sample1

}

obj1

}

o/p:

b value: true running disp()

k value: 12 running test()

1. **What is inheritance why we need it?**

**Ans.**

* A class acquiring/gathering property of another class is called as inheritance.
* The classes from where the properties are inherited are known as base class or Super class.
* The class through which properties are inherited is known as derived class or Sub class.
* Always subclass inherits the non-static properties from the super class.
* The static properties will never be inherited to the sub class.
* Whenever we create the instance of sub class that instance will always have the non-static properties of sub class and its super class

When we need to use the properties of another class and also create some own function or method at own class at that time you need to inheritance.

Ex,

class Demo1 { int k=12; void test() { System.out.println("runing test() of Demo1 "); } } class Sample1 extends D1{ //Sample1 inherit Demo1 double d=45.67; void disp() { System.out.println("running disp() of Sample1"); } }

class MainClass1 { public static void main(String[] args) {

//create a reference and with that we can call properties of Sample1 as well as Demo1 Sample1 ref1=new Sample1(); System.out.println("d value: "+ref1.d); ref1.disp();

System.out.println("K value: "+ref1.k); ref1.test(););//we can call k value or test () because of Sample1 have properties of Demo1 } }

o/p:

d value: 45.67 running disp() of Sample1

K value: 12 runing test() of Demo1

1. **What are the type of inheritance ,why doesn’t java support multiple inheritance**

**Ans.**

1. Single Inheritance: In this type of inheritance a sub class inherits properties from only one super class.
2. Multi-level inheritance: In this case the sub class inherits the properties of super class which inherits the properties of another super class. We can define any level of inheritance
3. Multiple Inheritance: Sub class inheriting from more than one super class is known as multiple inheritance java does not support multiple inheritance through classes.
4. Hierarchical Inheritance: In this type of inheritance more than one subclass inherits the properties from one super class, In other words the sub classes having common super class. This type of inheritance is used to achieve generalization.

**Why java doesn’t support multiple Inheritances?**

* Sub class constructor can’t call more than one super class because multiple “super ()” statements are not allowed.
* The multiple inheritances lead to the ambiguity of diamond problem because super class properties can’t be inherited to some sub class in two different classes.

**Object**

**Diamond**

Super();

Class B

Class A

Class C

1. **Difference between “super()” statement and “this()” statement?**

**Ans.**

**“super ()” statements:**

* “super ()” statements used to make a call to the super class constructor from sub class constructor.
* “super ()” statements should be used only inside the constructor body.
* “super ()” statements should be the first statements of constructor body; it can’t be used anywhere else.
* Multiple super statements is not allowed in the constructor.

Constructor of a class can call another constructor of same class are using “this()” statements

**“this()” statements:**

* “this()” statements can be used call constructor of the current class, it can’t call the constructor of super class
* “this ()”statements can be used to call method 0 arg or parameterize const.
* “this ()”statements should be used only In the constructor body and it must be the first statements of the constructor. Multiple this statements are not allowed
* Recursive constructor calls are not allowed.

see the example of question no 8..

1. **How memory are using in JVM?**

To execute any program the JVM makes use of following memory area:-

1. **Heap Area:**

The Heap Area is used to store the instance in the program. In the heap area the memory allocated is random.

The new operator loads the non-static member into heap area.

1. **Static Pool Area:**

This area is used for storing the static member of the class .The pool will be created for each class, the class loader program of JVM is responsible to load the static member of the class to the static position.

1. **Method Area:**

The method area is used to store the definition statements of methods.

1. **Stack Area:** The stack area is used for execution purpose and normal a statement to which has to be executed in JVM should be come to stack area.

The local variable components always store in stack memory area.

**STATIC POOL AREA**

**(**Static Member**)**

Storage

**STACK AREA**

**(**Execution**)**

**METHOD AREA**

**(**Methods**)**

Storage

**HEAP AREA**

**(**Non-Static**)**

Storage

**JVM**

**JVM**

Garbage collector

**HEAP AREA (**non-static member**)**

Inst of class2-obj1 Inst of class3/2-obj2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Identifier** | **Value** |  | **Identifier** | **Value** |
| int i | 3 | int j | 5 |
| disp(); |  |  |  |
|  |  |  |

Temporary value Temporary value

e.g.

|  |
| --- |
| disp(); |
| test(); |
| Main()  obj1.i address |

**STACK AREA (**Execution**)**

Called

Caller

**METHOD AREA/BODY(**static + non-static**)**

Main () st test () disp()

**STATIC POOL AREA (**only static member**)**

Main Class Class1

Static main()

St int k=2;

St test();

Method body

Method body

Method body

1. **Difference between blocks ,constructor, and methods**

Answer:

1. **Constructor** is used to initialize the state of object, whereas **method** is expose the behavior of object and whereas **blocks** are executed something.

2. **Constructor** and **blocks** must not have return type whereas **method** must have return type.

3. **Constructor** names same as the class name whereas **method** may or may not the same class name and whereas **blocks** have only static keyword and braces and for the non-static blocks there have only braces.

4. **Constructor and blocks** invoke implicitly whereas **method** invoke explicitly.

5. **Constructor** compiler provide default constructor whereas **method** compiler doesn’t provide. **Bocks** compile same as constructor but different is only static block and non-static blocks.

1. **Explain Final data members how to initialize it?**

Answer:

1. **Final keyword** can be applied to member variable, local variable, method or [class in Java](http://javarevisited.blogspot.com/2011/10/class-in-java-programming-general.html).
2. **Final member variable** must be initialized at the time of declaration or inside constructor, failure to do so will result in compilation error.
3. You cannot reassign value to *final variable in Java*.
4. **Local final variable** must be initializing during declaration.
5. Only final variable is accessible inside anonymous class in Java.
6. **Final method** cannot be [overridden in Java](http://javarevisited.blogspot.com/2011/12/method-overloading-vs-method-overriding.html).
7. **Final class** cannot be inheritable in Java.