3.3.3 Constructors

- Constructors are the special methods whose name is same as class name.
- Constructors do not have return type even void also.
- Constructors will be invoked by the JVM automatically at the time of object creation.
- Constructors are mainly used to initialize instance variables of class with different set of values.

Lab287.java

```
class Hello{
int a;
int b;
int c;

void show(){
System.out.println(a);
System.out.println(b);
System.out.println(c);
}
}
class Lab287{
public static void main(String args[]){
Hello h1=new Hello();
h1.show();

Hello h2=new Hello();
h2.show();

Hello h3=new Hello();
h3.show();

}
}
```

Lab288.java

```
class Hello{
int a;
int b;
int c;

void show(){
System.out.println(a+"\t"+b+"\t"+c);
}
}

class Lab288{
public static void main(String args[]){
Hello h1=new Hello();
h1.a=10;
h1.b=20;
h1.c=30;

h1.show();

Hello h2=new Hello();
h2.a=100;
h2.b=200;
h2.c=300;

h2.show();

Hello h3=new Hello();
h3.a=99;
h3.b=88;
h3.c=77;

h3.show();

}
}
```

Lab289.java

```
class Hello{
int a;
int b;
int c;

void init(int x,int y,int z){
a=x;
b=y;
c=z;
}
void show(){
System.out.println(a+"\t"+b+"\t"+c);
}
}
class Lab289{
public static void main(String args[]){
Hello h1=new Hello();
h1.init(10,20,30);
h1.show();

Hello h2=new Hello();
h2.init(100,200,300);
h2.show();

Hello h3=new Hello();
h3.init(99,88,77);
h3.show();
}
}
```

Lab290.java

```
class Hello{
int a;
int b;
int c;

Hello(int x,int y,int z){
a=x;
b=y;
c=z;
}
void show(){
System.out.println(a+"\t"+b+"\t"+c);
}
}

class Lab290{
public static void main(String args[]){
Hello h1=new Hello(10,20,30);
h1.show();

Hello h2=new Hello(100,200,300);
h2.show();

Hello h3=new Hello(99,88,77);
h3.show();

}
}
```

Lab291.java

```
class Hello{
int a;

{
System.out.println("Instance Block");
System.out.println(a);
}

Hello(){
System.out.println("0 arg - Con");
}

Hello(int x){
System.out.println("1 arg - Con");
a=x;
}
```

Lab292.java

```
class Hello{
int a;

{
System.out.println("Instance Block");
System.out.println(a);
}

Hello(){
System.out.println("0 arg - Con");
}

void show(){
System.out.println("show() method");
System.out.println(a);
}
}
```

```
void show(){
System.out.println("show() method");
System.out.println(a);
}
}

class Lab291{
public static void main(String args[]){

Hello h1=new Hello();
h1.show();

Hello h2=new Hello(99);
h2.show();
}
}
```

```
class Lab292{
public static void main(String args[]){

Hello h=new Hello(99);
h.show();

}
}
```

Lab293.java

```
class Hello{

Hello(){
System.out.println("0 arg - Con");
}

void show(){
System.out.println("show() method");
}
}

class Lab293{
public static void main(String args[]){

Hello h=new Hello();
h.show();

}
}
```

Lab294.java

```
class Hello{
int a;

Hello(){
System.out.println("0 arg - Con");
}

Hello(int x){
System.out.println("1 arg - Con");
a=x;
}

void show(){
System.out.println("show() method");
System.out.println(a);
}
}

class Lab294{
public static void main(String args[]){

Hello h=new Hello();
h.show();
h.Hello(99);
}
}
```

Lab295.java

```
class Hello{
int a;

Hello(){
System.out.println("0 arg - Con");
}

void Hello(int x){
System.out.println("1 arg -Method");
a=x;
}

void show(){
System.out.println("show() method");
System.out.println(a);
}
}

class Lab295{
public static void main(String args[]){

Hello h=new Hello();
h.show();
h.Hello(99);
}
}
```

Lab296.java

```
class Hello{
int a;

void Hello(int x){
System.out.println("1 arg -Method");
a=x;
}

void show(){
System.out.println("show() method");
System.out.println(a);
}
}

class Lab296{
public static void main(String args[]){
Hello h=new Hello(99);
h.show();
}
}
```

Lab297.java

```
class Customer{
int cid;
String cname;
String email;
long phone;
String city;

Customer(){
System.out.println("0 arg - con");
}

Customer(int cid,String cname,String email){
System.out.println("3 arg - con");
this.cid=cid;
this.cname=cname;
this.email=email;
}
```

```
Customer(int cid,String cname,String email,long phone){
System.out.println("4 arg - con");
this.cid=cid;
this.cname=cname;
this.email=email;
this.phone=phone;
}

Customer(int cid,String cname,String email,long phone,String city){
System.out.println("5 arg - con");
this.cid=cid;
this.cname=cname;
this.email=email;
this.phone=phone;
this.city=city;
}

void show(){
System.out.println(cid+"\t"+cname+"\t"+email+"\t"+phone+"\t"+city);
}
}

class Lab297{
public static void main(String args[]){
Customer cust1=new Customer();
cust1.show();

Customer cust2=new Customer(101,"Sri","sri@myjlc.com");
cust2.show();

Customer cust3=new Customer(102,"Vas","vas@myjlc.com",12345);
cust3.show();

Customer cust4=new Customer(103,"Srinivas","srinivass@myjlc.com",99999,"Blore");
cust4.show();

}
}
```

3.3.4 this keyword

- this is a keyword which acts as a reference variable.
- this reference variable contains address of current object.
- this is an instance reference variable and cannot be accessed from static context.
- this keyword can be used in three ways:
- Consider the following Hello class

```
class Hello{
int a;
int b;
void m1(){ ... }
void m2(){ ... }
Hello(int a, int b){ ... }
Hello(int a){ ... }
Hello(){ ... }
}
```

1) To access the variables

Syntax:

```
this.<variableName>
```

Ex:

```
this.a
this.b
```

2) To access the methods

Syntax:

```
this.<methodName>();
```

Ex:

```
this.m1();
this.m2();
```

3) To access the overloaded constructor

Syntax:

```
this (params);
```

Ex:

```
this()          -> Invokes Default Constructor
this(99)        -> Invokes 1-Arg Constructor
this(99,88)     -> Invokes 2-Arg Constructor
```

Lab298.java

```
class Hello{
int a=10;
void show(){
int a=20;
System.out.println(a); // Local
System.out.println(a); // Instance
}
}
class Lab298{
public static void main(String args[]){
Hello h=new Hello();
h.show();
}
}
```

Lab299.java

```
class Hello{
int a=10;
void show(){
int a=20;
System.out.println(a); // Local
System.out.println(this.a); // Instance
}
}
class Lab299{
public static void main(String args[]){
Hello h=new Hello();
h.show();
}
}
```

Lab300.java

```
class Hello{
int a=10;
void show(){
System.out.println(a); // Instance
System.out.println(this.a); // Instance
}
}
class Lab300{
public static void main(String args[]){
Hello h=new Hello();
h.show();
}
}
```

Lab301.java

```
class Hello{
int a=10;
static int b=20;

void show(){
String a="Hello";
String b="Hai";

System.out.println(a); // Local
System.out.println(b); // Local
System.out.println(this.a); // Instance
System.out.println(this.b); // Static
System.out.println(Hello.b); // Static
}
}

class Lab301{
public static void main(String args[]){
Hello h=new Hello();
h.show();
}
}
```

Lab302.java

```
class Hello{
int a;
int b;
int c;

Hello(int a,int b,int c){
a=a;
b=b;
c=c;
}
void show(){
System.out.println(a);
System.out.println(b);
System.out.println(c);
}
}
class Lab302{
public static void main(String args[]){
Hello h=new Hello(10,20,30);
h.show();
}
}
```

Lab303.java

```
class Hello{
int a;
int b;
int c;

Hello(int a,int b,int c){
this.a=a;
this.b=b;
this.c=c;
}
void show(){
System.out.println(this.a);
System.out.println(this.b);
System.out.println(this.c);
}
}
class Lab303{
public static void main(String args[]){
Hello h=new Hello(10,20,30);
h.show();
}
}
```

Lab304.java

```
class Hello{
int a;

Hello(){
System.out.println("Hello- 0 arg");
}
Hello(int a){
System.out.println("Hello- 1 arg");
this.a=a;
this();
}
void show(){
System.out.println(a);
}
}

class Lab304{
public static void main(String args[]){
Hello h=new Hello(10);
h.show();
}
}
```

Lab305.java

```
class Hello{
int a;

Hello(){
System.out.println("Hello- 0 arg");
}
Hello(int a){
this();
System.out.println("Hello- 1 arg");
this.a=a;
}
void show(){
System.out.println(a);
}
}

class Lab305{
public static void main(String args[]){
Hello h=new Hello(10);
h.show();
}
}
```

Lab306.java

```
class Hello{

Hello(){
this();
System.out.println("Hello- 0 arg");
}

}

class Lab306{
public static void main(String args[]){
Hello h=new Hello();
}
}
```

Lab307.java

```
class Hello{
int a;

Hello(){
this(10);
System.out.println("Hello- 0 arg");
}

Hello(int a){
this();
System.out.println("Hello- 0 arg");
}
}

class Lab307{
public static void main(String args[]){
Hello h=new Hello(99);
}
}
```

Lab308.java

```
class Hello{
int a;

Hello(){
System.out.println("Hello- 0 arg");
this(10);
}

Hello(int a){
System.out.println("Hello- 0 arg");
this.a=a;
}
}
class Lab308{
public static void main(String args[]){
Hello h=new Hello();
}
}
```

Lab309.java

```
class Hello{
static int a=10;

static void show(){
int a=20;
System.out.println(a);
System.out.println(a);
}

}
class Lab309{
public static void main(String args[]){
Hello h=new Hello();
h.show();
}
}
```

Lab310.java

```
class Hello{
static int a=10;

static void show(){
int a=20;
System.out.println(a);
System.out.println(this.a);
}
}

class Lab310{
public static void main(String args[]){
Hello h=new Hello();
h.show();
}
}
```

Lab311.java

```
class Hello{
void show(){
int a=20;
System.out.println(a);
System.out.println(this.a);
}
}

class Lab311{
public static void main(String args[]){
Hello h=new Hello();
h.show();
}
}
```

SUMMARY

Constructors:

1) The Constructor without any arguments is called as Default Constructor.
2) When you are not writing any constructor inside the class then one default constructor will be inserted by the Java Compiler.
3) When you are writing any constructor inside the class then default constructor will not be inserted by the Java Compiler.
4) You can write multiple constructors inside the class by changing parameters. This process is called as CONSTRUCTOR OVERLOADING.
5) When you are overloading constructors then parameters must differ in terms of:
    a. Number of parameters
    b. Type of parameters
    c. Order of parameters

```
class Hello{
int a;
String str;
Hello(int a,String str){ ... }  // VALID
Hello(int a,String str){ ... }  // INVALID
Hello(int a,int b){ ... }       // VALID
Hello(String str,int a){ ... } // VALID
}
```

6) You cannot invoke the constructor. Always JVM invokes automatically.
7) You cannot specify the return type for the constructor. When you specify the return type for the constructor then it will be treated as normal method.
8) You can use empty return statement inside the constructor.

this keyword:

9) You can use same name for local variables and instance variables.
10) When you access any variable directly then following things will happen:
    a. Checks whether that variable is declared in the local scope or not.
    b. If found in the local scope, that local variable will be used.
    c. If not found in the local scope then checks whether that variable is declared in the class scope or not.
    d. If found, that class level variable will be used.
11) When you have same name for local variables and class level variables then do the following:
    a. Refer the local variable directly.
    b. Refer the class level variable using this keyword.
12) call to constructor using this must be from constructor only, not from methods and blocks.

```
class Hello{
Hello(){}
Hello(int a){
this(); // VALID
}
{
this(); // INVALID
}
void show(){
this(); // INVALID
}
}
```

13) call to this must be first statement in constructor.
14) The process of invoking one constructor from another constructor using this is called as CONSTRUCTOR CHAINING.
15) this is an instance reference variable and cannot be referenced from static context.
16) Local variables cannot be referred using this keyword.

17) this is a final reference variable and cannot be modified.

a. this = null; // INVALID

b. this = new Hello(); // INVALID

18) this can be assigned to current class reference variable.

```
class Hello{
void show(){
Hello h1=this; // VALID
Hai h2=this;    // INVALID
}
}
```

```
class Hai{
void show(){
Hello h1=this; // INVALID
Hai h2=this;    // VALID
}
}
```

19) this keyword can be referred with current class name.

```
class Hello{
int a;
void show(){
System.out.println(Hello.this.a);          // VALID
System.out.println(Hai.this.a);  // INVALID
} }
class Hai{
int a;
}
```

20) Scope of Local variables starts from its declaration statement and it goes upto end of the block where it is declared.

21) Local variables must be accessed from or after declaration statement.

22) Class loading is the process of reading the required .class file and storing that information in the memory.

23) The class will be loaded in the memory by the JVM when first time the member of the class will be used by JVM.

a. When you are executing the class with java.exe then class will be loaded and static block will be executed.

```
java Hello
  class Hello{
  static{
  System.out.println("STATIC BLOCK");
  }
  public static void main(String str[]){}
  }
```

From java 7 if you don't have main method then class won't be loaded and static block won't be executed.

b. When you create the object of the class and before this statement no other member of the class is used then class will be loaded.

```
class Lab{
public static void main(String args[]){
new Hello();
new Hello();
}
}
```

c. When you access the static members of the class and before this statement no other member of the class is used then class will be loaded.

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

d. When you access the final static variables that is initialized in the same statement then JVM won't execute the static block.

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

```
class Hello{
final static int a=10;
static{
System.out.println("ST Block");
}
}
```

e. When you define the reference variable but not using any members of the class then class will not be loaded.

```
class Lab{
public static void main(String args[]){
Hello h=null;
}
}
```

Assignment #8

Q1) What is Constructor?

Q2) Can I specify return type of Constructor?

Q3) What will happen when I specify the return type for the constructor?

Q4) When will Constructor be invoked?

Q5) Can I use return statement inside the constructor?

Q6) Can I invoke Constructor explicitly?

Q7) What is Default Constructor?

Q8) What is Constructor overloading?

Q9) What is the rule of Constructor overloading?

Q10) What is Constructor chaining?

Q11) What is the use of this keyword?

Q12) Why we cannot use this keyword in static block?

Q13) How many ways this keyword can be used?

Q14) Can I access local variable using this keyword?

Q15) Can I assign current class object to this reference variable?

Q16) Can I assign null to this reference variable?

Q17) Can I access this keyword with class name?

Q18) Can I take same name for local variables and class level variables?

Q19) What are the checks perform by JVM when I access any variable directly?

Q20) Can I write Static block inside the constructor?

Practice Test #8

Q.No	Question	Options	Answer

1	class Test1{
	public static void main(String args[]){
	Hello h=new Hello();
	}
	}
	class Hello{
	int a;
	Hello(){
	System.out.println("Hello Cons");
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Hello Cons
		D) Nothing will print	

2	class Test2{
	public static void main(String args[]){
	Hello h=new Hello();
	}
	}
	class Hello{
	int a;
	void Hello(){
	System.out.println("Hello Cons");
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Hello Cons
		D) Nothing will print	

3	class Test3{
	public static void main(String args[]){
	Hello h=new Hello();
	}
	}
	class Hello{
	int a;
	Hello(){
	System.out.println("Hello Cons");
	return;
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Hello Cons
		D) Hello Cons
		0
		E) Nothing will print	

4	class Test4{
	public static void main(String args[]){
	Hello h=new Hello(10);
	System.out.println(h.a);
	}
	}
	class Hello{
	int a;
	Hello(int a){
	System.out.println("Hello Cons");
	this.a=a;
	return;
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Hello Cons
		D) Hello Cons
		10
		E) Nothing will print	

5	class Test5{
	public static void main(String args[]){
	Hello h=new Hello(10);
	System.out.println(h.a);
	}
	}
	class Hello{
	int a;
	Hello(int a){
	System.out.println("Hello Cons");
	a=a;
	return;
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Hello Cons
		D) Hello Cons
		0
		E) Nothing will print	

6	class Test6{
	public static void main(String args[]){
	Hello h=new Hello();
	}
	}
	class Hello{
	int a;
	Hello(int a){
	System.out.println("Hello Cons");
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Hello Cons
		D) Nothing will print	

7	Class Test7{
	public static void main(String args[]){
	Hello h=new Hello(10);
	} }
	class Hello{
	int a;
	Hello(int a){
	this();
	System.out.println("Hello Cons");
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Hello Cons
		D) Nothing will print	

8	class Test8{
	public static void main(String args[]){
	Hello h=new Hello();
	}
	}
	class Hello{
	int a;
	Hello(){
	this(10);
	}
	Hello(int a){
	System.out.println("Hello Cons");
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Hello Cons
		D) Nothing will print	

9	class Test9{
	public static void main(String args[]){
	Hello h=new Hello();
	h.Hello(10);
	} }
	class Hello{
	int a;
	Hello(){
	System.out.println("Hello Cons");
	}
	void Hello(int a){
	System.out.println("void Hello Cons");
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) void Hello Cons
		Hello Cons
		D) Hello Cons
		void Hello Cons
		E) Nothing will print	

10	class Test10{
	public static void main(String args[]){
	Hello h=new Hello(12);
	}
	}
	class Hello{
	int a=10;
	Hello(int a){
	System.out.println(a);
	System.out.println(this.a);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) 10
		12
		D) 12
		10
		E) Nothing will print	

11	class Test11{
	public static void main(String args[]){
	Hello h=new Hello(12);
	System.out.println("OK");
	}
	}
	class Hello{
	Hello(int a){
	System.out.println(this.a);
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) 11
		OK
		D) 0
		OK
		E) Nothing will print	

12	class Test12{
	public static void main(String args[]){
	Hello h=new Hello(12);
	}
	}
	class Hello{
	Hello(){
	System.out.println("Hello 1");
	}
	Hello(int a){
	Hello();
	System.out.println("Hello 2");
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) Hello 1
		Hello 2
		D) Hello 2
		Hello 1
		E) Nothing will print	

13	class Test13{
	public static void main(String args[]){
	System.out.println(Hello.a);
	}
	}
	
	class Hello{
	static int a=10;
	Hello(){
	System.out.println("CONS");
	}
	static{
	System.out.println("SB");
	}
	}
		A) Compilation Error
		B) Runtime Error
		C) SB
		CONS
		10
		D) SB
		10
		E) CONS
		SB
		10	