Note::
 When an object of parent class is created,the parent class constructor will be called.
 When an object of child class is created,both parent and child class constructor will be called because of super().

```
class Parent
{
	//Constructor
	Parent()
	{
		System.out.println("Parent class constructor");
	}
}
class Child extends Parent
{
	//Constructor
	Child()
	{
		System.out.println("Child class constructor");
	}
}

public class Test
{
	public static void main(String[] args)
	{
		Parent p = new Parent();
		System.out.println();
		Child c= new Child();
	}
}
```

Output
```
Parent class constructor

Parent class constructor
Child class constructor
```

Explain the importance of super()?
 In case of Parent class, if parent class has a parameterized constructor then in child class constructor compulsorily there should
 be a call to parent class parameterized constructor otherwise the code would result in "CompileTime Error".

super vs super()
++++++++++++++++
 super() -> it is used to call parent class constructor from the child class.
 super   -> It is used to avoid name clash of properties and behaviours b/w parent and child class.

eg#1.
```
class Person
{
	String name;
	int age;
	String gender;
	float height;
```

```
	//Parameterized constructor :: shadowing
	Person(String name,int age,String gender,float height)
	{
		System.out.println("Person class constructor");
		this.name   = name;
		this.age    = age;
		this.gender = gender;
		this.height = height;
	}
	public void dispDetails()
	{
		System.out.println("Name   is :: "+name);
		System.out.println("Height is :: "+height);
		System.out.println("Age    is :: "+age);
		System.out.println("Gender is :: "+gender);
	}
}

class Student extends Person
{
	String sid;
	int marks;

	//Parameterized constructor :shadowing
	Student(String name,int age, String gender, float height, String sid,int
marks)
	{
		//call to parent class parameterized constructor
		super(name,age,gender,height);
		System.out.println("Student class constructor");
		this.sid = sid;
		this.marks = marks;
	}

	public void dispDetails()
	{
		//To call parent class dispDetails() we used super keyword.
		super.dispDetails();
		System.out.println("SID   ::   "+sid);
		System.out.println("MARKS ::   "+marks);
	}
}

public class Test
{
	public static void main(String[] args)
	{
		Student std = new Student("sachin",49,"M",5.4f,"IND10",35);
		std.dispDetails();
	}
}

Output
Person class constructor
Student class constructor
Name   is :: sachin
Height is :: 5.4
Age    is :: 49
```

```
Gender is :: M
SID   ::   IND10
MARKS ::   35
```

Note:
Explain the importance of this keyword vs super keyword?
 this : It is used to avoid shadowing problem[name clash b/w local variable and instance variable]
 super : It is used to avoid name clash b/w properties and behaviours of parent and child class[Inheritance]

eg#1.

```
class A
{
	int i;
	A()
	{
		System.out.println("Parent class constructor");
	}
}
class B extends A
{
	int i;

	B()
	{
		System.out.println("Child class constructor");
	}

	public void setData(int x, int i)
	{
		//Giving x value to parent class i
		super.i = x;

		//Giving i value to child class i
		this.i = i;
	}

	public void disp()
	{
		System.out.println("A class i value is :: " +super.i);
		System.out.println("B class i value is :: " +i);
	}
}

public class Test
{
	public static void main(String[] args)
	{
		B b = new B();
		b.setData(10,20);
		b.disp();
	}
}
```

Output

```
A class i value is :: 10
B class i value is :: 20
```

Constructor Overloading
++++++++++++++++++++++
A class can contain more than one constructor,this way of writing a constructor we call it as "Constructor Overloading".
When we have constructor overloading in our code, to make a call to constructor within a class we use "this()".

super() : it will take the control to parent class constructor.
eg#1.

```
//Constructor Overloading
class Demo
{
	Demo(int i)
	{
		super();
		System.out.println("int arg constructor");
	}

	Demo(float f)
	{
		super();
		System.out.println("float arg constructor");
	}

	Demo()
	{
		super();
		System.out.println("zero arg constructor");
	}
}

public class Test
{
	public static void main(String[] args)
	{
		Demo d1= new Demo();
		Demo d2= new Demo(10);
		Demo d3= new Demo(10.5f);
	}
}
```

Output

```
zero arg constructor
int arg constructor
float arg constructor
```

eg#2.

```
//Constructor Overloading
this() :: It is used to make a call to current class constructors only.

class Demo
{
	Demo(int i)
	{
		this(10.5f);
		System.out.println("int arg constructor");
	}

	Demo(float f)
```

```
	{
		System.out.println("float arg constructor");
	}

	Demo()
	{
		this(10);
		System.out.println("zero arg constructor");
	}
}

public class Test
{
	public static void main(String[] args)
	{
		Demo d1= new Demo();//float-arg/int-arg/zero-arg constructor
		System.out.println();

		Demo d2= new Demo(10);//float-arg/int-arg constructor
		System.out.println();

		Demo d3= new Demo(10.5f);//float-arg constructor
	}
}

Output
float arg constructor
int arg constructor
zero arg constructor

float arg constructor
int arg constructor

float arg constructor

Predict the output of the following code from the compiler
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
1.
class Test
{

}

Compiler
class Test exteds Object
{
	Test()
	{
		super();
	}
}

2.
public class Test
{

}

Compiler
```

```
public class Test exteds Object
{
	public Test()
	{
		super();
	}
}


3.
class Test
{
	void test()
	{

	}

}

Compiler
class Test exteds Object
{
	Test()
	{
		super();
	}
	void test()
	{

	}
}

4.
class Test
{
	Test(int i)
	{

	}

}

Compiler
class Test exteds Object
{
	Test(int i )
	{
		super();
	}
}

5.
class Test
{
	Test(int i)
	{
		this();
	}
	Test()
```

```
	{

	}
}
Compiler
class Test extends Object
{
	Test(int i)
	{
		this();
	}
	Test()
	{
		super();
	}
}
```