

```

1 package com.pack1;
2
3 public class ClassA
4 {
5     ClassA(int x)
6     {
7
8     }
9
10 }
11
12
13

```

```

1 package com.pack1;
2
3 public class ClassB extends ClassA
4 {
5
6
7
8 }
9
10
11
12
13

```

1. In class B compiler will automatically provide a default constructor which is having `super()` as its first statement
2. As in class A there is no default constructor, from class B default constructor `super()` is trying to call Class A default constructor we will be getting a compiler time error.
3. To overcome this problem, the programmer should explicitly call the parent class constructor.
4. whenever we are using the inheritance concept 100% before child class constructor parent class constructor should be executed otherwise, we are getting a compile time error.

Here, we are getting an error because in class B we have a default constructor which is provided by the compiler. In that default constructor the first statement by default is `super` constructor call (`super ()`) and compiler is trying to call

parent class default constructor. Which is not available here so that is the reason we are getting an error.

<pre>1 package com.pack1; 2 3 public class ClassA 4 { 5 6 } 7 8</pre>	<pre>1 package com.pack1; 2 3 public class ClassB extends ClassA 4 { 5 6 } 7 8</pre>
<pre>1 package com.pack1; 2 3 public class ClassA 4 { 5 ClassA() 6 { 7 8 } 9 }</pre>	<pre>1 package com.pack1; 2 3 public class ClassB extends ClassA 4 { 5 6 } 7 8 9</pre>
<pre>1 package com.pack1; 2 3 public class ClassA 4 { 5 ClassA(int x) 6 { 7 8 } 9 }</pre>	<pre>1 package com.pack1; 2 3 public class ClassB extends ClassA 4 { 5 ClassB() 6 { 7 super(100); 8 } 9 }</pre>
<pre>1 package com.pack1; 2 3 public class ClassA 4 { 5 ClassA(int x) 6 { 7 8 } 9 } 10 11 12 13 14 15 16</pre>	<pre>1 package com.pack1; 2 3 public class ClassB extends ClassA 4 { 5 ClassB() 6 { 7 8 } 9 ClassB(int x) 10 { 11 super(100); 12 } 13 } 14 15 16</pre>

```

1 package com.pack1;
2
3 public class ClassA
4 {
5     ClassA()
6     {
7
8     }
9     ClassA(int x)
10    {
11
12    }
13 }
14

```

```

1 package com.pack1;
2
3 public class ClassB extends ClassA
4 {
5     ClassB()
6     {
7
8     }
9     ClassB(int x)
10    {
11        this();
12        super(100);
13    }
14 }

```

```

1 package com.pack1;
2
3 public class ClassA
4 {
5     ClassA()
6     {
7
8     }
9     ClassA(int x)
10    {
11
12    }
13 }
14
15

```

```

1 package com.pack1;
2
3 public class ClassB extends ClassA
4 {
5     ClassB()
6     {
7
8     }
9     ClassB(int x)
10    {
11        super(100);
12        this();
13    }
14 }
15

```

```

1 package com.pack1;
2
3 public class ClassA
4 {
5     void meth1()
6     {
7
8     }
9     ClassA()
10    {
11
12    }
13     ClassA(int x)
14    {
15
16    }
17 }
18
19
20
21

```

```

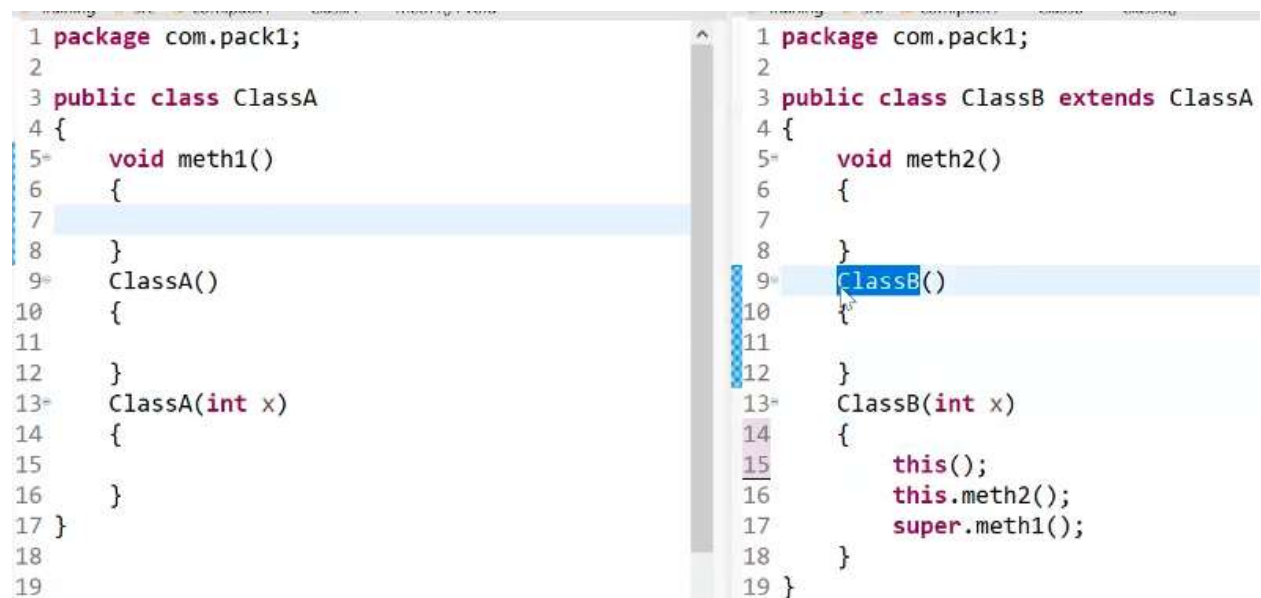
1 package com.pack1;
2
3 public class ClassB extends ClassA
4 {
5     void meth2()
6     {
7
8     }
9     ClassB()
10    {
11
12    }
13     ClassB(int x)
14    {
15        super(100);
16        //this(); // C.E
17
18        this.meth2();
19        super.meth1();
20    }
21 }

```

Note

****whenever we are performing inheritance inside the child class constructor we cannot write both super() and this() because both should be the 1st statement inside a constructor.

We can write super keyword and this keyword inside a constructor. (we can not use these key words inside a static area)



```
1 package com.pack1;
2
3 public class ClassA
4 {
5     void meth1()
6     {
7
8     }
9     ClassA()
10    {
11
12    }
13    ClassA(int x)
14    {
15
16    }
17 }
18
19
```

```
1 package com.pack1;
2
3 public class ClassB extends ClassA
4 {
5     void meth2()
6     {
7
8     }
9     ClassB()
10    {
11
12    }
13    ClassB(int x)
14    {
15        this();
16        this.meth2();
17        super.meth1();
18    }
19 }
```

```

1 package com.pack1;
2
3 public class ClassA
4 {
5     void meth1()
6     {
7
8     }
9     ClassA()
10    {
11
12    }
13    ClassA(int x)
14    {
15
16    }
17 }
18
19

```

```

1 package com.pack1;
2
3 public class ClassB extends ClassA
4 {
5     void meth2()
6     {
7
8     }
9     ClassB()
10    {
11        this(100);
12    }
13    ClassB(int x)
14    {
15        this();
16        this.meth2();
17        super.meth1();
18    }
19 }

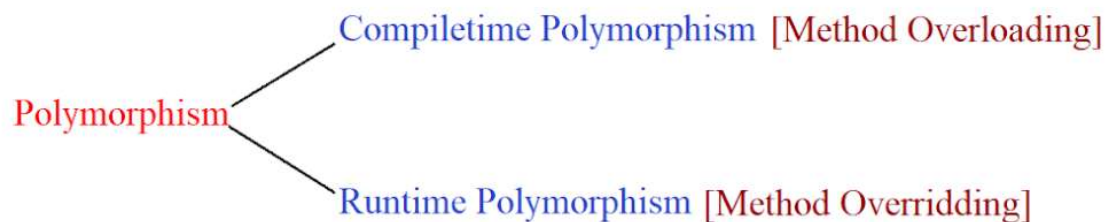
```

If we have first statement `this()` then here `super ()` will not be executed in this case but , if constructor is not having any `this()` as first statement then `super()` will be executed.

`this()` must be first statement and `super()` also must be first statement in constructor.

Constructors do not participate in inheritance but here I have a small question. But whenever we are creating an object for child class, parent class constructor also getting executed because of super constructor call (`super()`). Literally we are calling the constructor that is actually not participating in inheritance. Parent class constructor will be executed only we are calling and calling is automatically done by `super()`. But without executing parent constructor Inheritance will not occur we will be getting an error.

Polymorphism: It is a process of performing multiple tasks with the same identity.



Method Overloading: "Writing two or more methods in the same class having same method name but different parameters."

Understanding Method Signature

- A method signature represents method name, method return type and its parameter.
- It is useful to uniquely identify different methods.
- JVM identifies a method separately when the following differences are found
 - Different number of parameters in both methods
 - Different data types of parameters in both methods
 - Different in the return type in both methods

Method Overloading

- “Writing two or more methods with the same name but with different method signature is called Method Overloading”.
- It is also known as ‘**early-binding**’ or ‘**compile-time**’ polymorphism .
- Method calls are resolved at compile time.

Rules for performing overloading :

- ✓ **Must have different argument lists.**
 - ✓ May have different return types, as long as the argument lists are also different.
 - ✓ May have different access modifiers.
 - ✓ May throw different exceptions.
-
- Constructor over loading is possible.(We can’t write constructors inside an interface).
 - Recursion in java is a process in which a method calls itself continuously, when we are trying to achieve recursion by using constructor overloading, we will be getting compile time error as “Recursive constructor invocation”.

- Can we overload main method? (Y/N)
- Can we declare overloaded method as final? (Y/N)
- Can we over load two methods if one method is static and other is non-static method? (Y/N)
- Can we achieve method overloading by changing the return -type? (Y/N)

```

1 package com.pack1;
2
3 public class ClassA
4 {
5     void meth1()
6     {
7
8     }
9     void meth1() I
10    {
11
12    }
13    void meth1()
14    {
15
16    }
17    void meth1()
18    {
19
20    }
21    void meth1()
22    {
23
24    }

```

```

1 package com.pack1;
2
3 public class ClassA
4 {
5     void meth1()
6     {
7
8     }
9     void meth1(int i)
10    {
11
12    }
13    void meth1(String s)
14    {
15
16    }
17    void meth1(int i, String s)
18    {
19
20    }
21    void meth1(String s, int i)
22    {
23

```

```

11     System.out.println(20);
12 }
13 void meth1(String s)
14 {
15     System.out.println(30);
16 }
17 void meth1(int i, String s)
18 {
19     System.out.println(40);
20 }
21 void meth1(String s, int i)
22 {
23     System.out.println(50);
24 }
25 void meth1(StringBuffer sb)
26 {
27     System.out.println(60);
28 }
29 void meth1(byte b)
30 {
31     System.out.println(70);
32 }
33 public static void main(String[] args)
34 {
35     ClassA aobj=new ClassA();

```

```

24 }
25 void meth1(StringBuffer sb)
26 {
27     System.out.println(60);
28 }
29 void meth1(byte b)
30 {
31     System.out.println(70);
32 }
33 public static void main(String[] args)
34 {
35     ClassA aobj=new ClassA();
36     aobj.meth1();
37     aobj.meth1(100);
38     aobj.meth1("Java");
39     aobj.meth1(100,"Java");
40     aobj.meth1("Java",100);
41     aobj.meth1("Java is awesome");
42     aobj.meth1(10);
43 }
44 }

```

10
 20
 30
 40
 50
 30
 20

SCP is allocated only for string class

Why because to pass string buffer and string builder we need to use new key word

For byte the number it takes default as integer in java.

```
22 {
23     System.out.println(50);
24 }
25 void meth1(StringBuffer sb)
26 {
27     System.out.println(60);
28 }
29 void meth1(byte b)
30 {
31     System.out.println(70);
32 }
33 public static void main(String[] args)
34 {
35     ClassA aobj=new ClassA();
36     aobj.meth1();
37     aobj.meth1(100);
38     aobj.meth1("Java");
39     aobj.meth1(100,"Java");
40     aobj.meth1("Java",100);
41     aobj.meth1(new StringBuffer("Java"));
42     aobj.meth1((byte)10);
43 }
44 }
```

In the above case all Access Modifiers are default

```

1 package com.pack1;
2
3 public class ClassA
4 {
5     public void meth1()
6     {
7         System.out.println(10);
8     }
9     void meth1(int i)
10    {
11        System.out.println(20);
12    }
13    private void meth1(String s)
14    {
15        System.out.println(30);
16    }
17    void meth1(int i, String s)
18    {
19        System.out.println(40);
20    }
21    protected void meth1(String s, int i)
22    {
23        System.out.println(50);
24    }

```

```

9     void meth1(int i)
10    {
11        System.out.println(20);
12    }
13    private void meth1(String s)
14    {
15        System.out.println(30);
16    }
17    static void meth1(int i, String s)
18    {
19        System.out.println(40);
20    }
21    protected void meth1(String s, int i)
22    {
23        System.out.println(50);
24    }
25    void meth1(StringBuffer sb)
26    {
27        System.out.println(60);
28    }
29    static private void meth1(byte b)
30    {
31        System.out.println(70);
32    }

```

```

31     System.out.println(70);
32 }
33 public static void main(String[] args)
34 {
35     ClassA aobj=new ClassA();
36     aobj.meth1();
37     aobj.meth1(100);
38     aobj.meth1("Java");
39     aobj.meth1(100,"Java");
40     aobj.meth1("Java",100);
41     aobj.meth1(new StringBuffer("Java"));
42     aobj.meth1((byte)10);
43     main();
44     main("Java is awesome");
45 }
46 public static void main()
47 {
48     System.out.println("1st main()");
49 }
50 public static void main(String s)
51 {
52     System.out.println("2nd main()");
53 }
54 }

```

```

48     System.out.println("1st main()");
49 }
50 public static void main(String s)
51 {
52     System.out.println("2nd main()");
53 }
54 ClassA()
55 {
56     this(100);
57     System.out.println("ClassA default constructor");
58 }
59 ClassA(int x)
60 {
61     System.out.println("ClassA Parameterized constructor : "+x);
62 }
63 }

```

We can overload

A public method

A private method

A protected method

A default method

A constructor

A main method

A abstract method

A final method

Any method in java we can overload.

```
2
3 public class ClassA
4 {
5     public static void meth1()
6     {
7         System.out.println(10);
8     }
9     void meth1(int i)
10    {
11        System.out.println(20);
12    }
13    private void meth1(String s)
14    {
15        System.out.println(30);
16    }
17    static void meth1(int i, String s)
18    {
19        System.out.println(40);
20    }
21    protected void meth1(String s, int i)
22    {
23        System.out.println(50);
24    }
```

```

25= void meth1(StringBuffer sb)
26 {
27     System.out.println(60);
28 }
29= static private void meth1(byte b)
30 {
31     System.out.println(70);
32 }
33= public static void main(String[] args)
34 {
35     ClassA aobj=new ClassA();
36     aobj.meth1();
37     aobj.meth1(100);
38     aobj.meth1("Java");
39     aobj.meth1(100,"Java");
40     aobj.meth1("Java",100);
41     aobj.meth1(new StringBuffer("Java"));
42     aobj.meth1((byte)10);
43     main();
44     main("Java is awesome");
45 }
46= public static void main()
47 {
48     System.out.println("1st main()");
49 }

50= public static void main(String s)
51 {
52     System.out.println("2nd main()");
53 }
54= ClassA()
55 {
56     this(100);
57     System.out.println("ClassA default constructor");
58 }
59= ClassA(int x)
60 {
61     System.out.println("ClassA Parameterized constructor : "+x);
62 }
63 }

```

Assignment

```
3 public class ClassA
4 {
5     void meth1()
6     {
7         System.out.println(10);
8         System.out.println(this.meth2()+56);
9     }
10    int meth2()
11    {
12        System.out.println(75);
13        System.out.println(96);
14        return this.meth3();
15    }
16    int meth3()
17    {
18        System.out.println(74);
19        System.out.println(92);
20        return 74-92;
21    }
22    void meth4()
23    {
24        System.out.println(56);
25    }
26    ClassA()
27    {
28        this(52);
29        this.meth1();
30        System.out.println(85);
31    }
32    ClassA(int a)
33    {
34        System.out.println(a+7);
35    }
36 }
```

```
3 public class ClassB extends ClassA
4 {
5     void display()
6     {
7         System.out.println("hi");
8         super.meth4();
9     }
10    static int show(int a)
11    {
12        System.out.println(a+a);
13        return a+a++;
14    }
15    ClassB()
16    {
17        this(show(50));
18        for(int i=1;i++)
19        {
20            super.meth4();
21            break;
22        }
23        System.out.println("hi");
24        System.out.println(show(50));
25    }
26    ClassB(int a)
27    {
28        System.out.println("==>"+(a+++show(50)));
29    }
30    public static void main(String[] args)
```

Class-B

```

3 public class ClassB extends ClassA
4 {
5     void display()
6     {
7         System.out.println("hi");
8         super.meth4();
9     }
10    static int show(int a)
11    {
12        System.out.println(a+a);
13        return a+a++;
14    }
15    ClassB()
16    {
17        this(show(50));
18        for(int i=1;;i++)
19        {
20            super.meth4();
21            break;
22        }
23        System.out.println("hi");
24        System.out.println(show(50));
25    }
26    ClassB(int a)
27    {
28        System.out.println("==>"+(a+++show(50)));
29    }

30    public static void main(String[] args)
31    {
32        new ClassB().display();
33    }
34 }

```