```
    {
        int count=1;
        while(true)
        {
            System.out.println("ClassA run() : "+count++);
        }
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();

        Thread t=new Thread(aobj);
        t.start();

        for(int i=1;i<=20;i++)
        {
            System.out.println("main() : "+i);
        }
    }
}
```

```
ClassA run() : 344901
ClassA run() : 344902
ClassA run() : 344903
ClassA run() : 344904
ClassA run() : 344905
ClassA run() : 344906
ClassA run() : 344907
ClassA run() : 344908
ClassA run() : 344909
ClassA run() : 344910
ClassA run() : 344911
ClassA run() : 344912
ClassA run() : 344913
ClassA run() : 344914
ClassA run() : 344915
ClassA run() : 344916
ClassA run() : 344917
ClassA run() : 344918
ClassA run() : 344919
ClassA run() : 344920
ClassA run() : 344921
```

```
public class ClassA extends Thread
{
    @Override
    public void run()
    {
        int count=1;
        while(true)
        {
            System.out.println("ClassA run() : "+count++);
        }
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();

        Thread t=new Thread(aobj);
        t.setDaemon(true);
        t.start();

        for(int i=1;i<=20;i++)
        {
            System.out.println("main() : "+i);
        }
```

```
main() : 7
main() : 8
main() : 9
main() : 10
main() : 11
main() : 12
main() : 13
main() : 14
main() : 15
main() : 16
main() : 17
ClassA run() : 69
ClassA run() : 70
ClassA run() : 71
ClassA run() : 72
ClassA run() : 73
ClassA run() : 74
main() : 18
ClassA run() : 75
ClassA run() : 76
main() : 19
main() : 20
```

Daemon Thread

A daemon thread in java is a type of thread that runs in the background and does not prevent the JVM from existing when the program finishes executing.

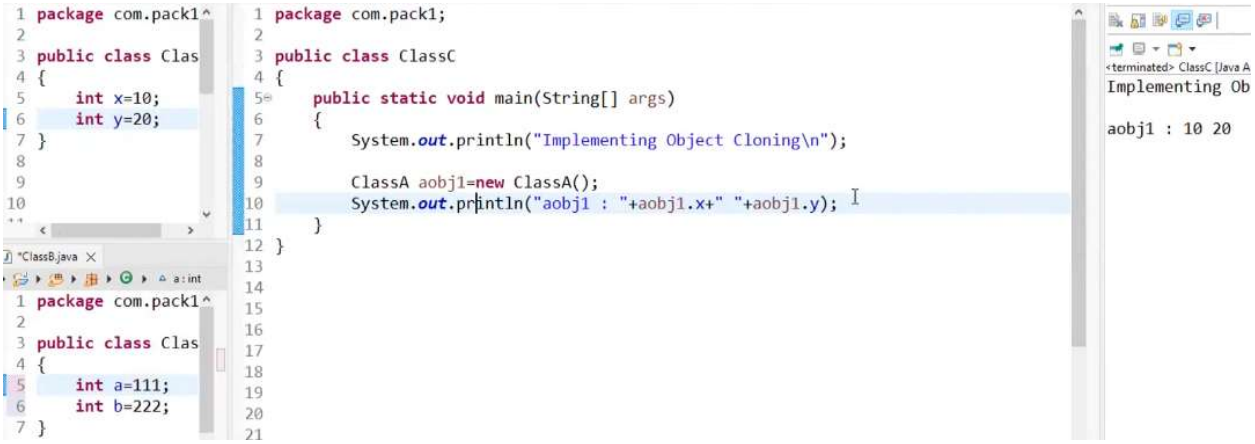
In other words when all the non-daemon threads in a program have finished their execution then JVM will automatically terminate Daemon thread.

A daemon thread is a low priory, infinite execution thread which supports in the background of a java program.

Example: Garbage collector.

1) What is a Thread?
2)Which Thread by default runs in every java program?
3) What is the default priority of the Thread?
4) How can you change the priority number of the Thread?
5) Which method is executed by any Thread?
6) How can you stop a Thread which is running?
7) Explain the two types of multitasking?
8) What is the difference between a process and a Thread?
9) What is Thread scheduler?
10) Explain the synchronization of Threads?
11) What is the difference between synchronized block and synchronized method?
12) What is Thread deadlock? How can you resolve deadlock situation?
13) Which methods are used in Thread communication?
14) What is the difference between notify() and notifyAll() methods?
15) What is the difference between sleep(100) and wait(100) methods and join()?
16) Explain the life cycle of a Thread?
17) What is daemon Thread?// Garbage collector
Def: Daemon thread is a low priority thread that runs in background to perform tasks such as garbage collection (gc) etc.,

clone(), Which is present in Object class



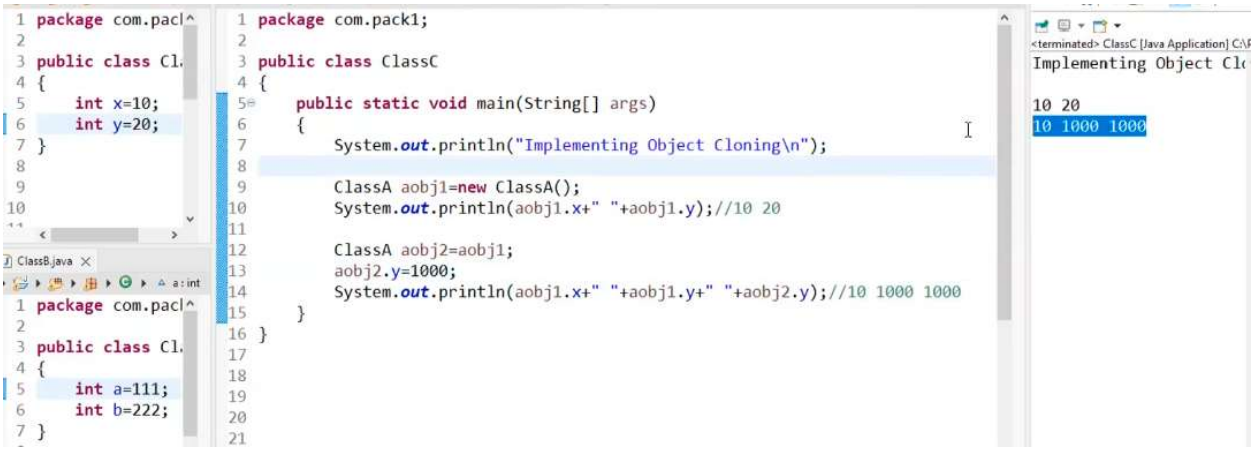
```
package com.pack1;

public class Clas
{
    int x=10;
    int y=20;
}
```

```
package com.pack1;

public class ClassC
{
    public static void main(String[] args)
    {
        System.out.println("Implementing Object Cloning\n");

        ClassA aobj1=new ClassA();
        System.out.println("aobj1 : "+aobj1.x+" "+aobj1.y);
    }
}
```

```
<terminated> ClassC [Java A
Implementing Ob

aobj1 : 10 20
```

```
package com.pack1;

public class Clas
{
    int a=111;
    int b=222;
}
```

This not cloning

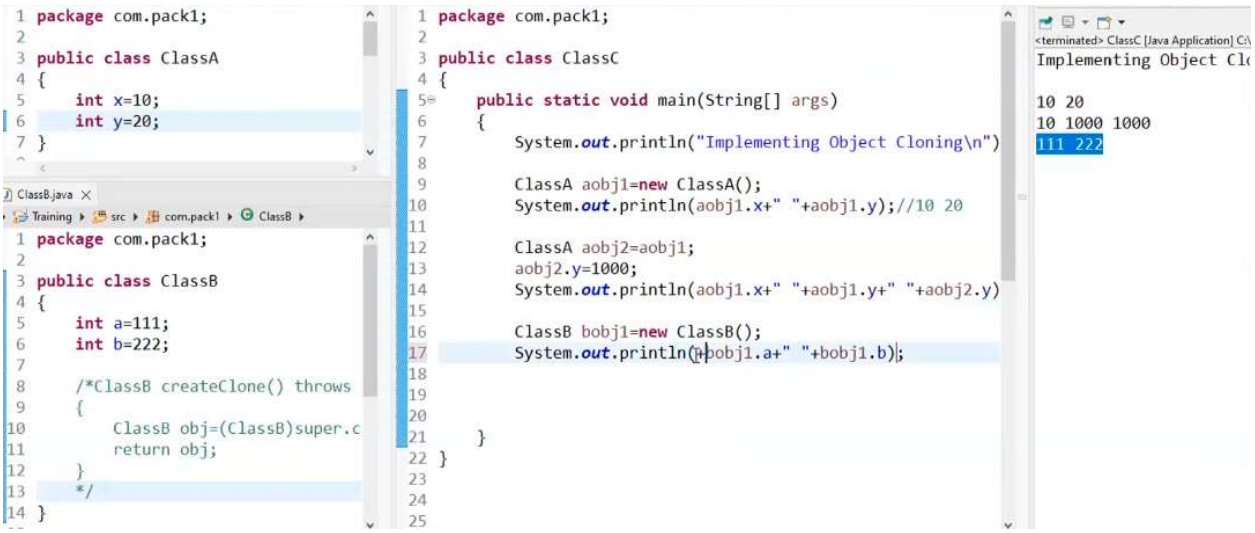


```
package com.pack1;

public class Cl
{
    int x=10;
    int y=20;
}
```

```
package com.pack1;

public class ClassC
{
    public static void main(String[] args)
    {
        System.out.println("Implementing Object Cloning\n");

        ClassA aobj1=new ClassA();
        System.out.println(aobj1.x+" "+aobj1.y);//10 20

        ClassA aobj2=aobj1;
        aobj2.y=1000;
        System.out.println(aobj1.x+" "+aobj1.y+" "+aobj2.y);//10 1000 1000
    }
}
```

```
<terminated> ClassC [Java Application] C:\
Implementing Object Clo

10 20
10 1000 1000
```

```
package com.pack1;

public class Cl
{
    int a=111;
    int b=222;
}
```

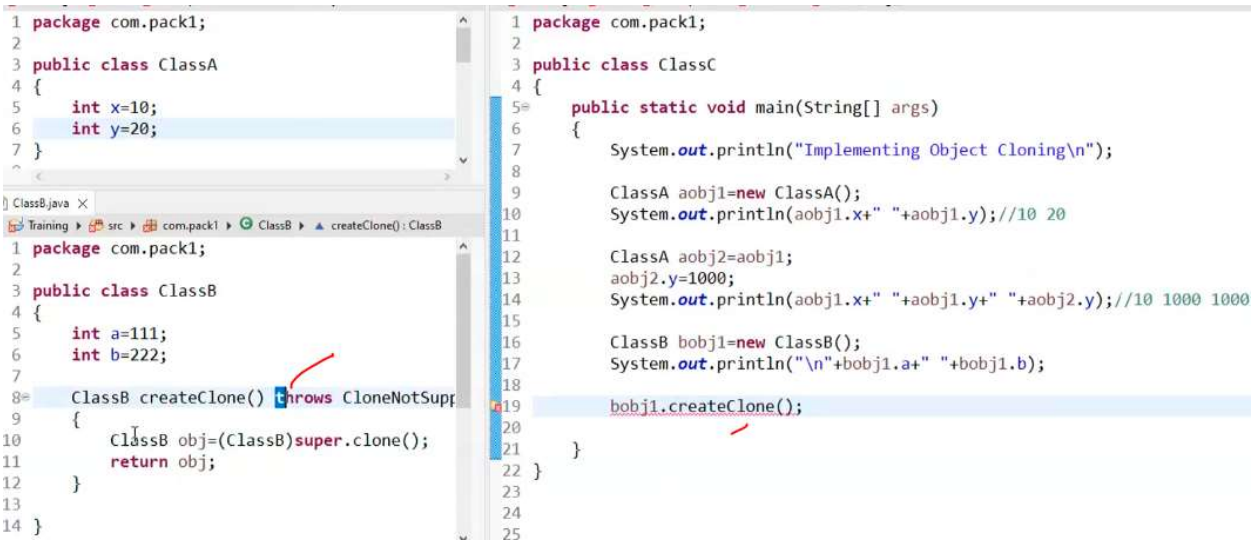


```
package com.pack1;

public class ClassA
{
    int x=10;
    int y=20;
}
```

```
package com.pack1;

public class ClassB
{
    int a=111;
    int b=222;

    /*ClassB createClone() throws
    {
        ClassB obj=(ClassB)super.c
        return obj;
    }
    */
}
```

```
package com.pack1;

public class ClassC
{
    public static void main(String[] args)
    {
        System.out.println("Implementing Object Cloning\n");

        ClassA aobj1=new ClassA();
        System.out.println(aobj1.x+" "+aobj1.y);//10 20

        ClassA aobj2=aobj1;
        aobj2.y=1000;
        System.out.println(aobj1.x+" "+aobj1.y+" "+aobj2.y)

        ClassB bobj1=new ClassB();
        System.out.println(bobj1.a+" "+bobj1.b);


    }
}
```

```
<terminated> ClassC [Java Application] C:\
Implementing Object Clo

10 20
10 1000 1000
111 222
```



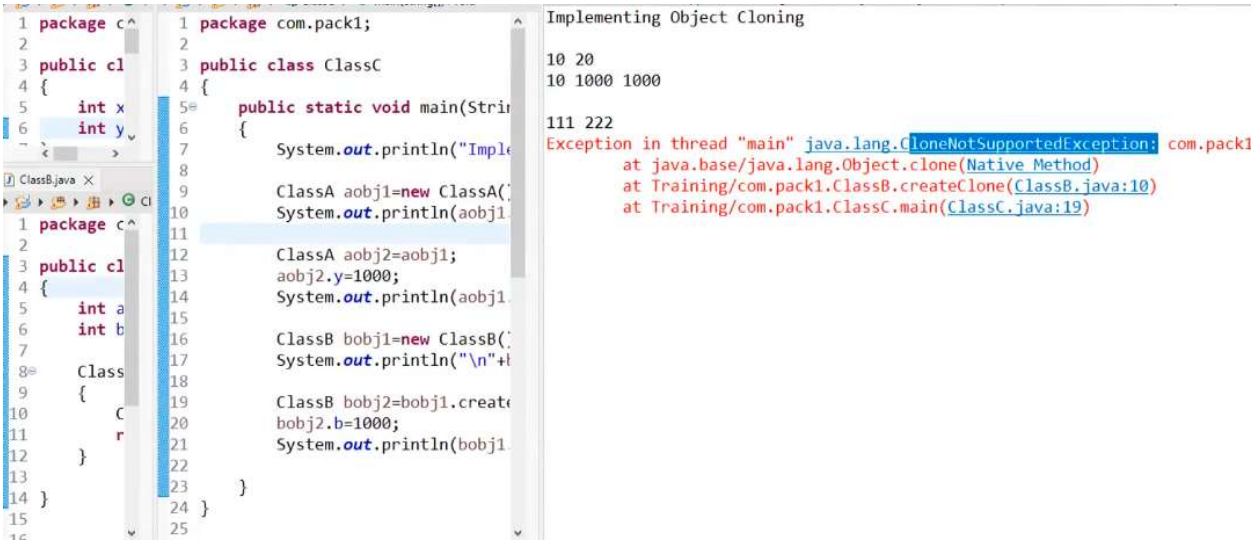
```
package com.pack1;

public class ClassA
{
    int x=10;
    int y=20;
}
```

ClassB.java

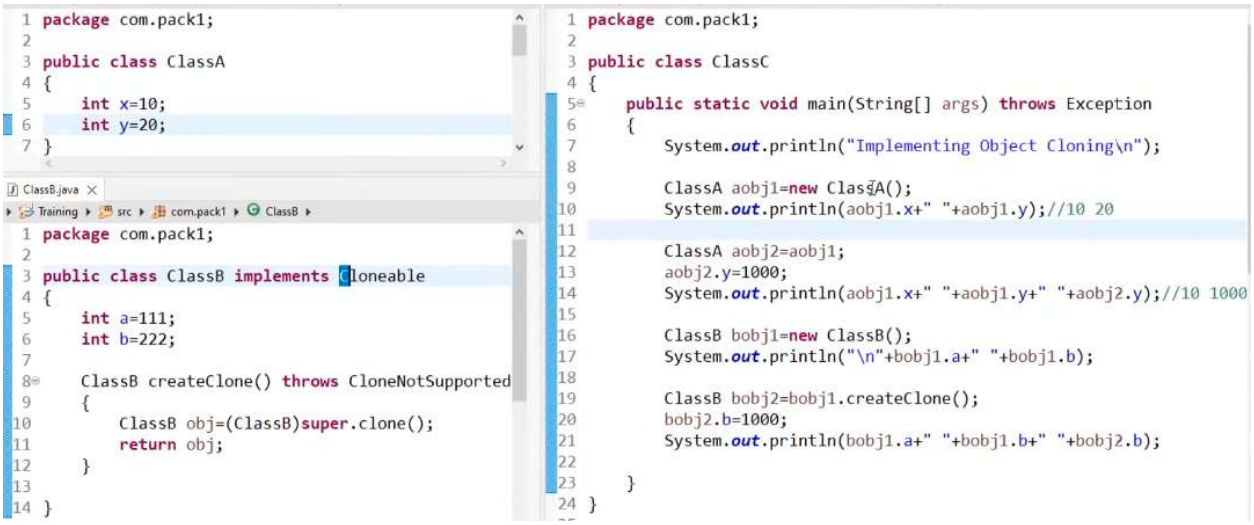
Training ▸ src ▸ com.pack1 ▸ ClassB ▸ createClone() : ClassB

```
package com.pack1;

public class ClassB
{
    int a=111;
    int b=222;

    ClassB createClone() throws CloneNotSupp
    {
        ClassB obj=(ClassB)super.clone();
        return obj;
    }

}
```

```
package com.pack1;

public class ClassC
{
    public static void main(String[] args)
    {
        System.out.println("Implementing Object Cloning\n");

        ClassA aobj1=new ClassA();
        System.out.println(aobj1.x+" "+aobj1.y);//10 20

        ClassA aobj2=aobj1;
        aobj2.y=1000;
        System.out.println(aobj1.x+" "+aobj1.y+" "+aobj2.y);//10 1000 1000

        ClassB bobj1=new ClassB();
        System.out.println("\n"+bobj1.a+" "+bobj1.b);

        bobj1.createClone();

    }
}
```



```
Implementing Object Cloning

10 20
10 1000 1000

111 222
Exception in thread "main" java.lang.CloneNotSupportedException: com.pack1
        at java.base/java.lang.Object.clone(Native Method)
        at Training/com.pack1.ClassB.createClone(ClassB.java:10)
        at Training/com.pack1.ClassC.main(ClassC.java:19)
```



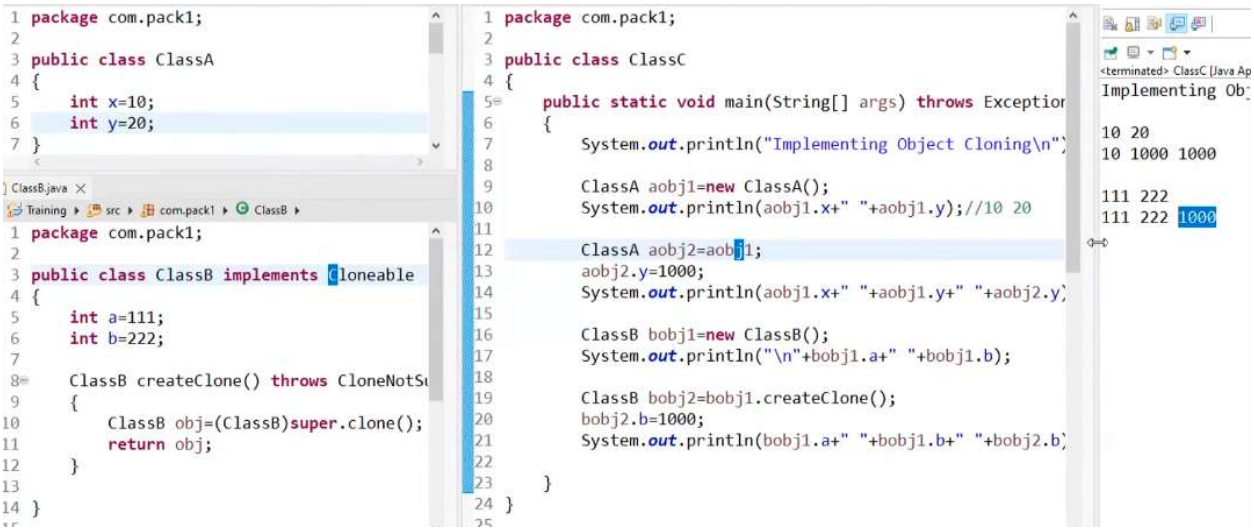
```
package com.pack1;

public class ClassA
{
    int x=10;
    int y=20;
}
```

ClassB.java

Training ▸ src ▸ com.pack1 ▸ ClassB ▸

```
package com.pack1;

public class ClassB implements Cloneable
{
    int a=111;
    int b=222;

    ClassB createClone() throws CloneNotSupported
    {
        ClassB obj=(ClassB)super.clone();
        return obj;
    }

}
```

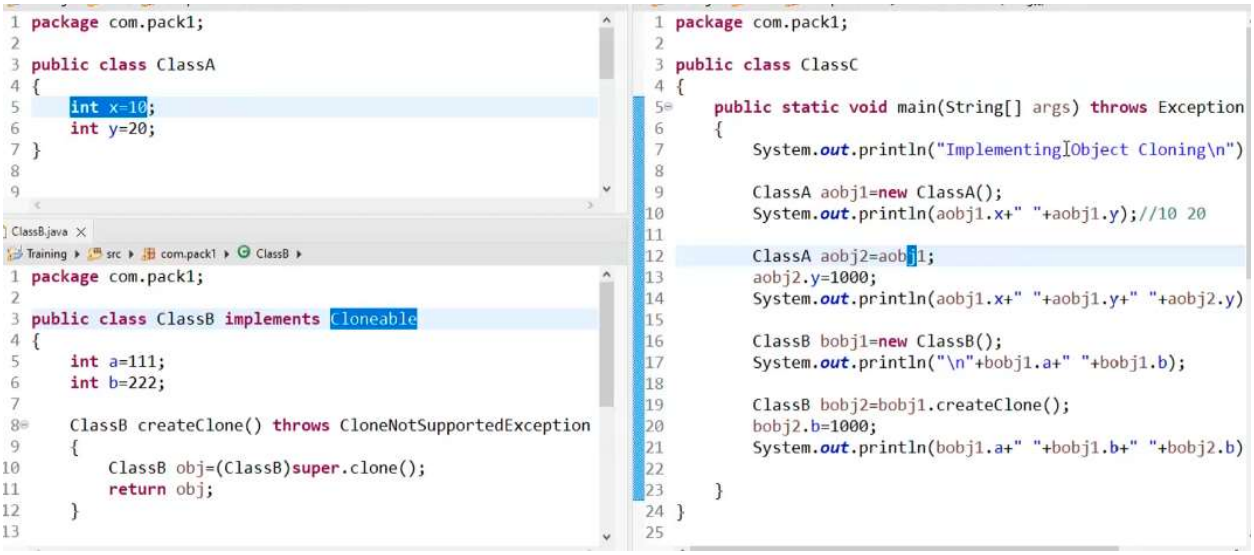
```
package com.pack1;

public class ClassC
{
    public static void main(String[] args) throws Exception
    {
        System.out.println("Implementing Object Cloning\n");

        ClassA aobj1=new ClassA();
        System.out.println(aobj1.x+" "+aobj1.y);//10 20

        ClassA aobj2=aobj1;
        aobj2.y=1000;
        System.out.println(aobj1.x+" "+aobj1.y+" "+aobj2.y);//10 1000

        ClassB bobj1=new ClassB();
        System.out.println("\n"+bobj1.a+" "+bobj1.b);

        ClassB bobj2=bobj1.createClone();
        bobj2.b=1000;
        System.out.println(bobj1.a+" "+bobj1.b+" "+bobj2.b);

    }
}
```



Understanding clone()

clone() is present in object class which is used to create an exact copy of an existing object without using new key word.

If we want to use clone() in a class, then 100% that class should be inheriting a marker interface (Empty interface) known as Cloneable Interface, otherwise we will be getting clone not supported exception.



```java
package com.pack1;

public class ClassC
{
    public static void main(String[] args) throws Exception
    {
        System.out.println("Implementing Object Cloning\n");

        ClassA aobj1=new ClassA();
        System.out.println(aobj1.x+" "+aobj1.y);//10 20

        ClassA aobj2=aobj1;
        aobj2.y=1000;
        System.out.println(aobj1.x+" "+aobj1.y+" "+aobj2.y);//10 1000 1000

        ClassB bobj1=new ClassB();
        System.out.println("\n"+bobj1.a+" "+bobj1.b);

        ClassB bobj2=bobj1.createClone();
        bobj2.b=1000;
        System.out.println(bobj1.a+" "+bobj1.b+" "+bobj2.b);

    }
}
```