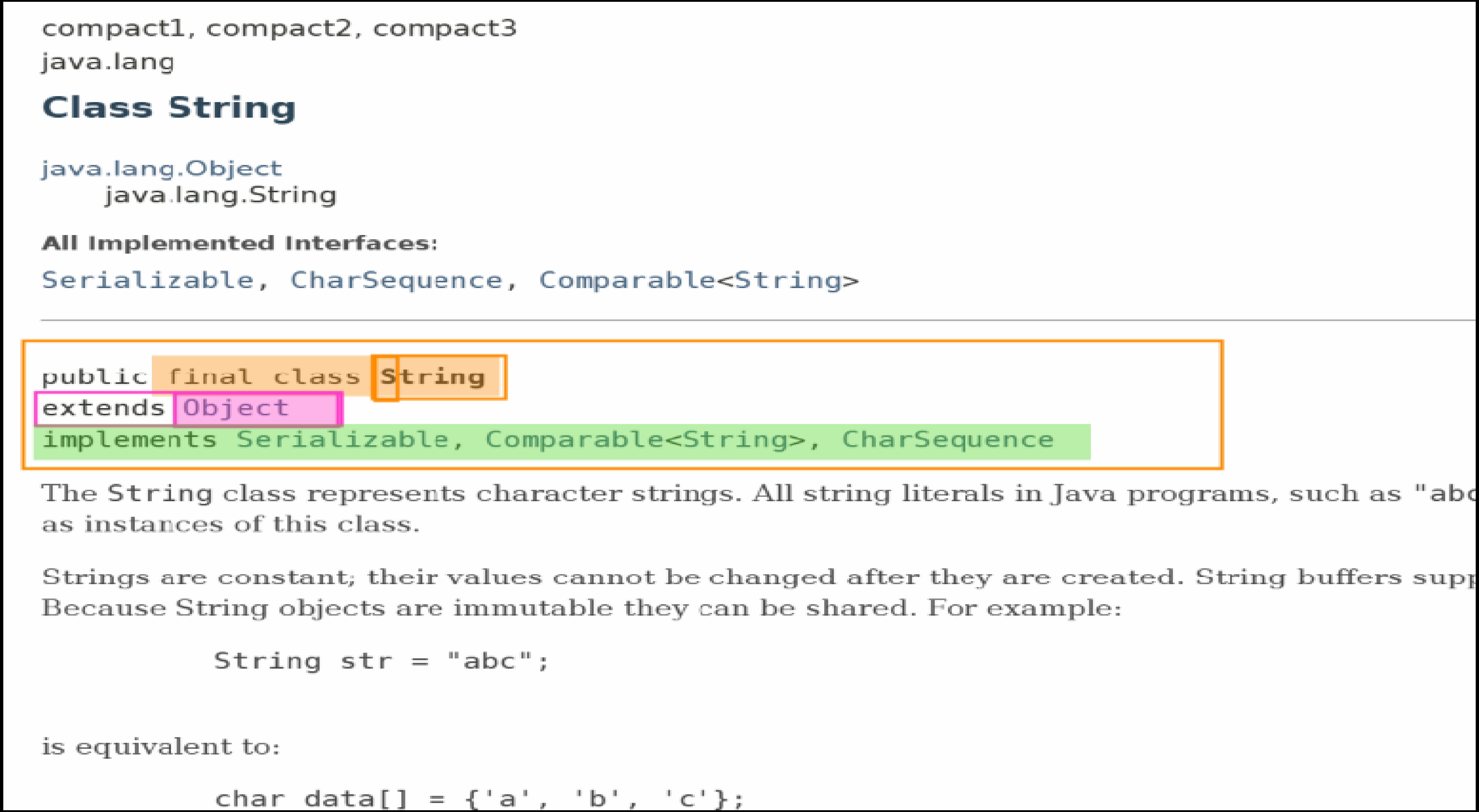
Object Oriented Programming with Java (OOPJ)

Session 11: String

Kiran Waghmare



compact1, compact2, compact3
java.lang

Class String

java.lang.Object
    java.lang.String

All Implemented Interfaces:

Serializable, CharSequence, Comparable<String>

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence
```

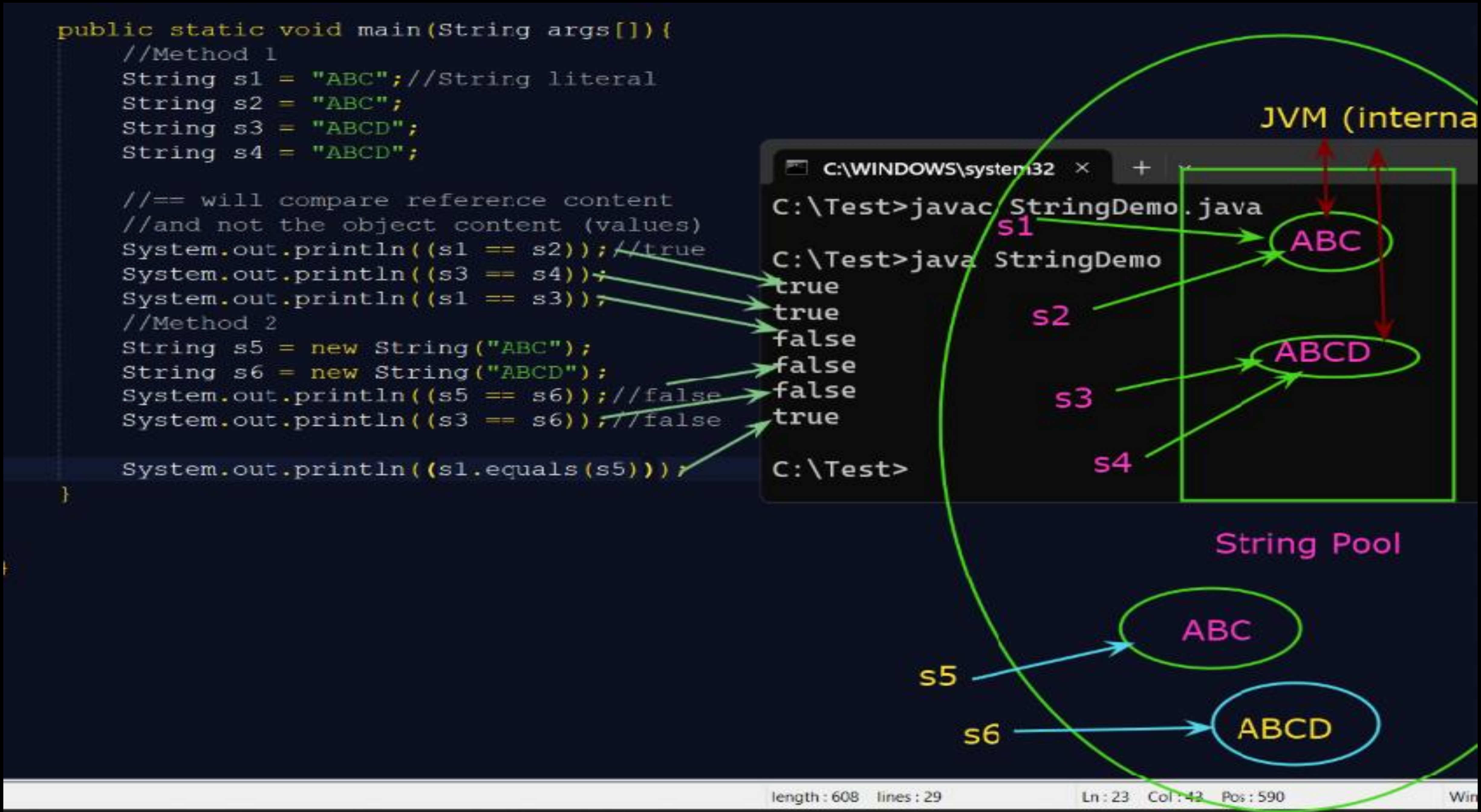
The `String` class represents character strings. All string literals in Java programs, such as "abc
as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers supp
Because String objects are immutable they can be shared. For example:

```
     String str = "abc";
```

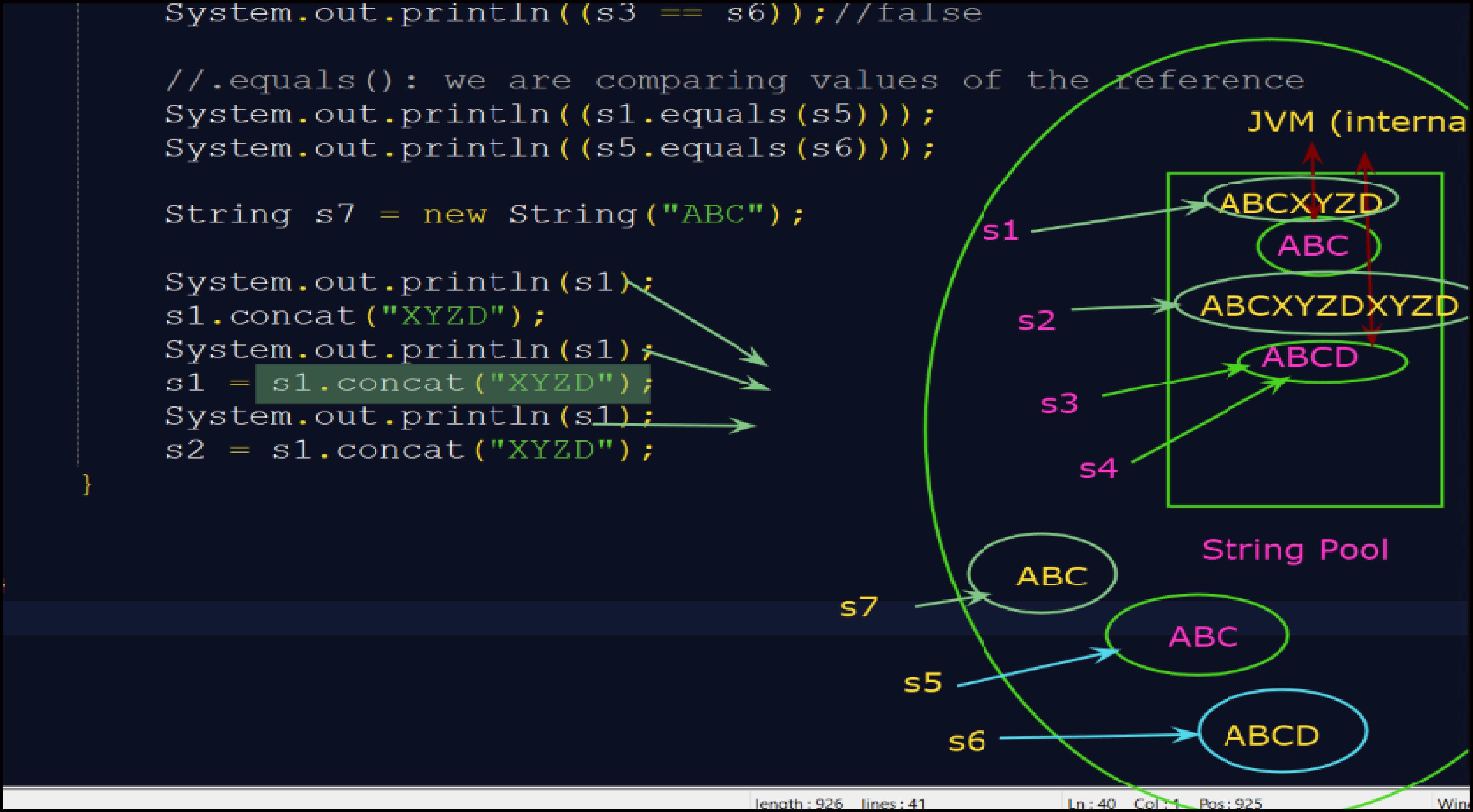
is equivalent to:

```
     char data[] = {'a', 'b', 'c'};
```



```java
public static void main(String args[]){
    //Method 1
    String s1 = "ABC";//String literal
    String s2 = "ABC";
    String s3 = "ABCD";
    String s4 = "ABCD";

    //== will compare reference content
    //and not the object content (values)
    System.out.println((s1 == s2));//true
    System.out.println((s3 == s4));
    System.out.println((s1 == s3));
    //Method 2
    String s5 = new String("ABC");
    String s6 = new String("ABCD");
    System.out.println((s5 == s6));//false
    System.out.println((s3 == s6));//false

    System.out.println((s1.equals(s5)));
}
```

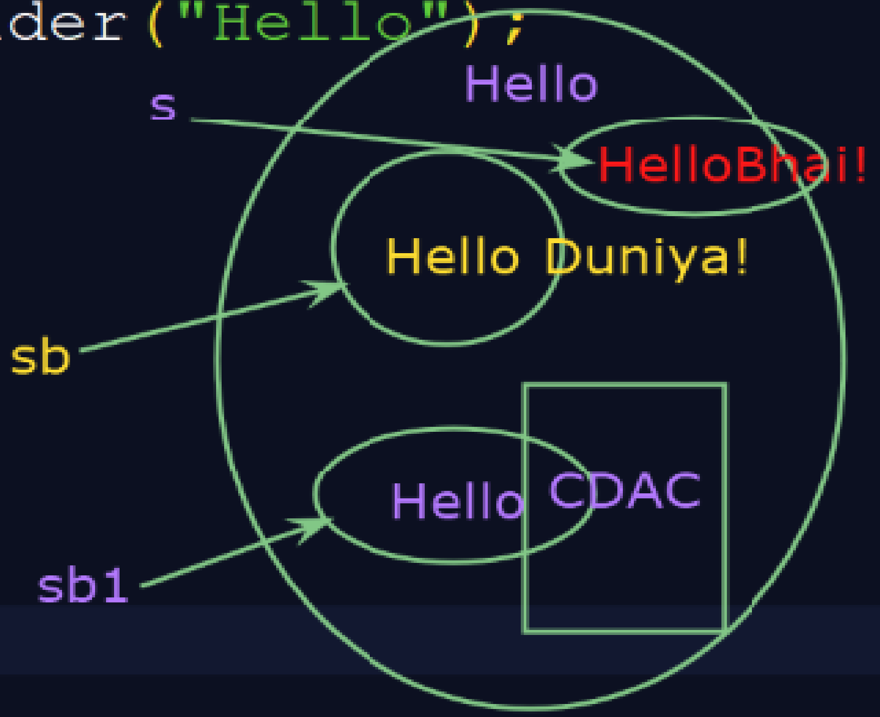
```
C:\Test>javac StringDemo.java

C:\Test>java StringDemo
true
true
false
false
false
true

C:\Test>
```



```
        System.out.println((s3 == s6));//false

        //.equals(): we are comparing values of the reference
        System.out.println((s1.equals(s5)));
        System.out.println((s5.equals(s6)));

        String s7 = new String("ABC");

        System.out.println(s1);
        s1.concat("XYZD");
        System.out.println(s1);
        s1 = s1.concat("XYZD");
        System.out.println(s1);
        s2 = s1.concat("XYZD");
}
```

length : 926   lines : 41   Ln : 40   Col : 1   Pos : 925   Win

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

StringBuffer sb = new StringBuffer("Hello");
System.out.println(sb);
sb.append("Duniya!");
System.out.println(sb);

StringBuilder sb1 = new StringBuilder("Hello");
System.out.println(sb1);
sb1.append("CDAC!");
System.out.println(sb1);

String s = "Hello";
System.out.println(s);
s.concat("Bhai!");
System.out.println(s);
s=s.concat("Bhai!");
System.out.println(s);
}
```



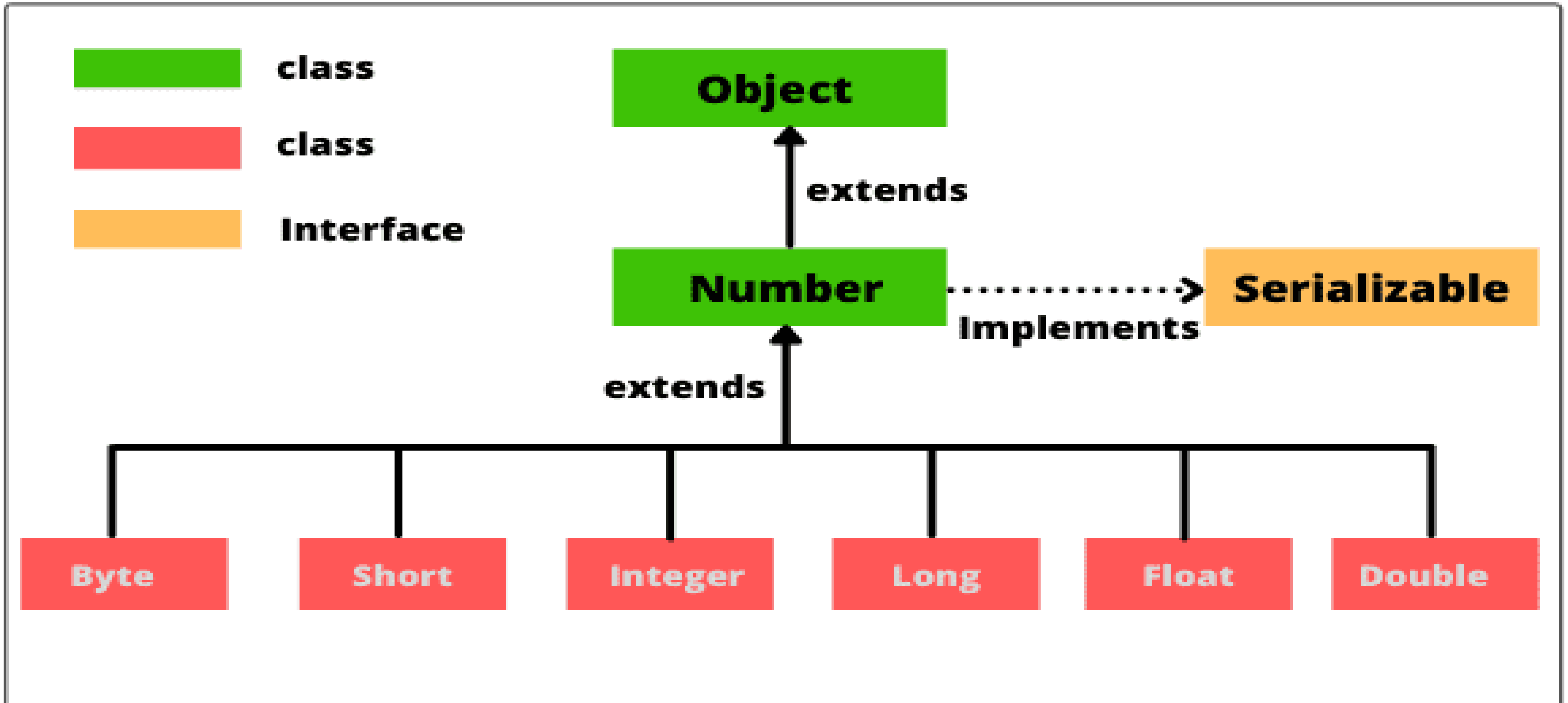
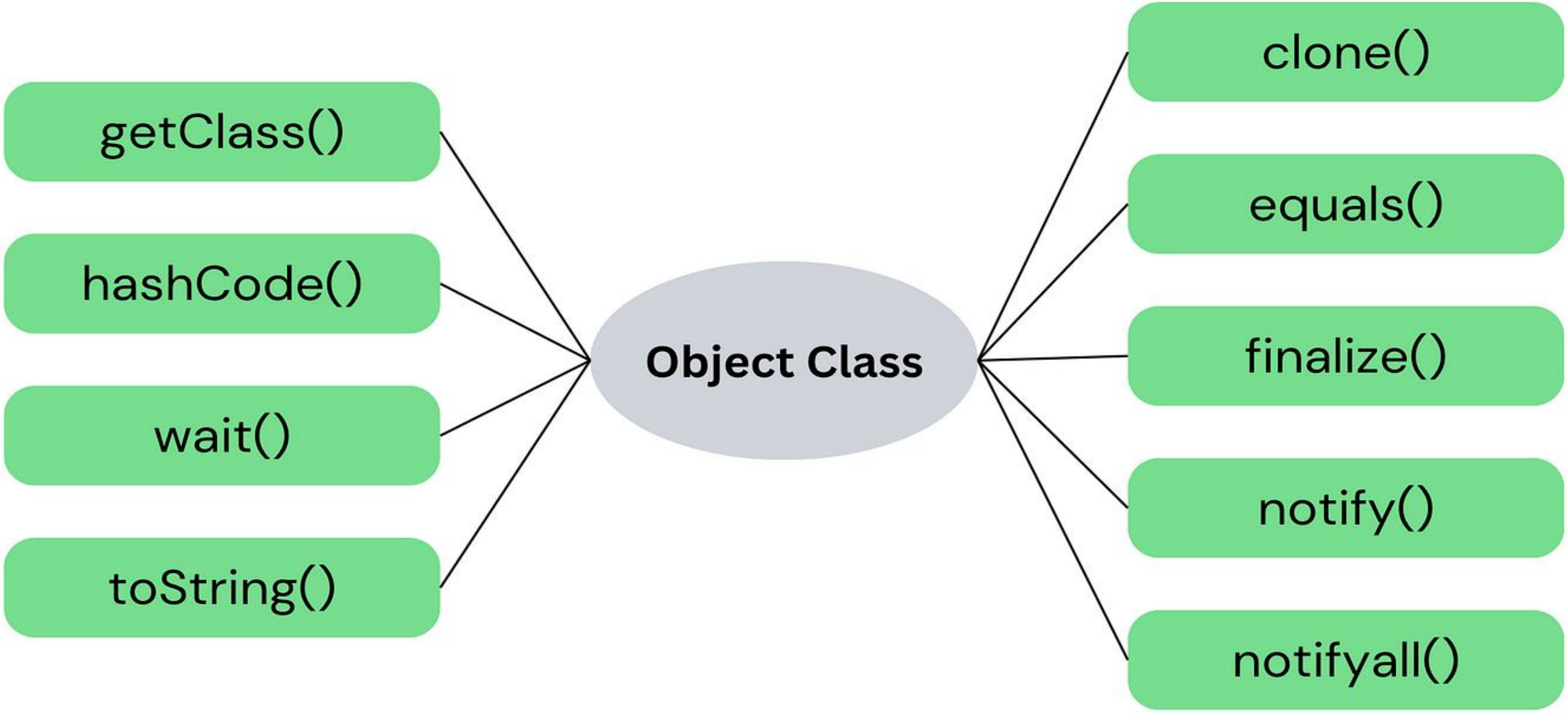
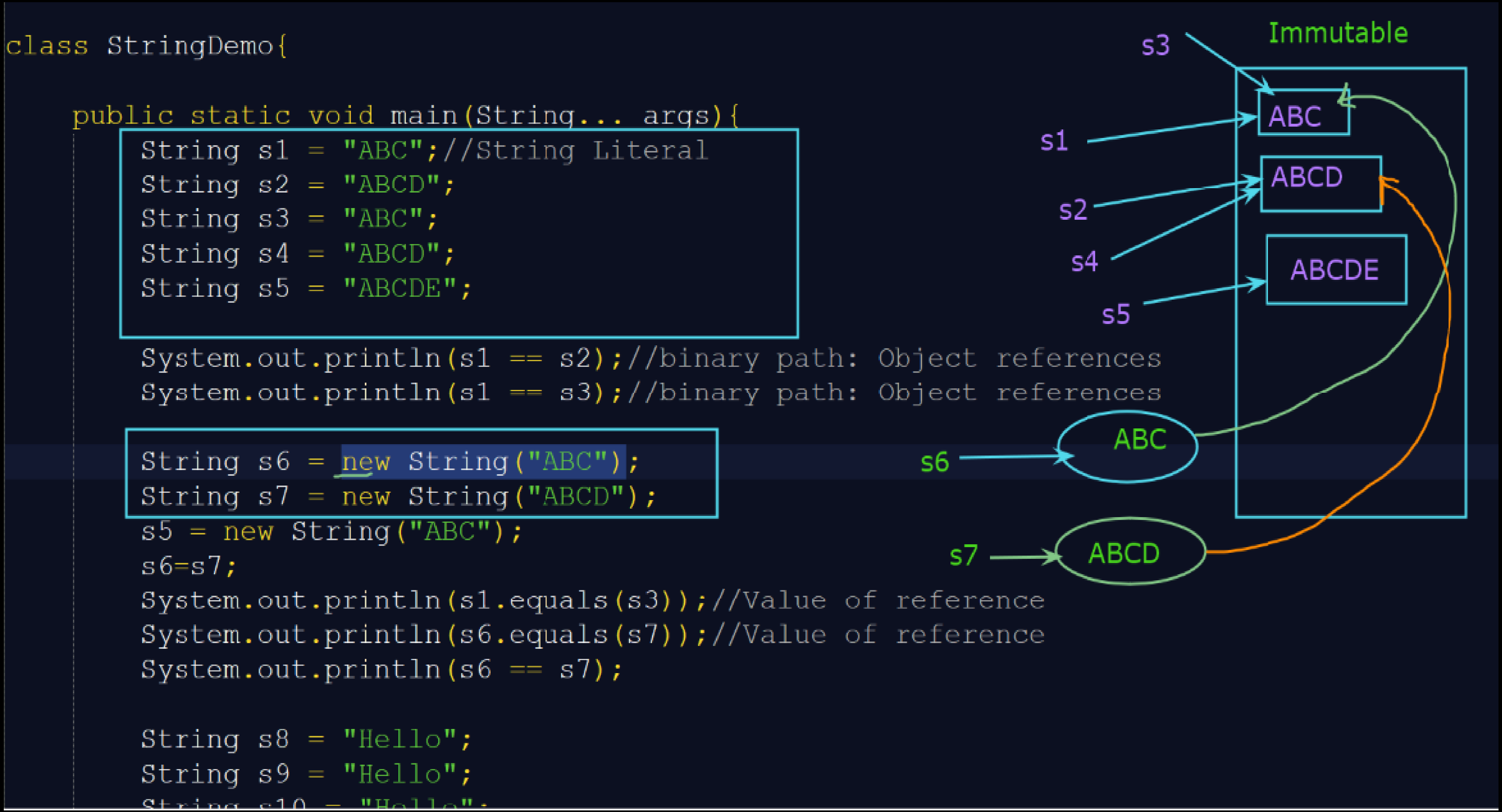
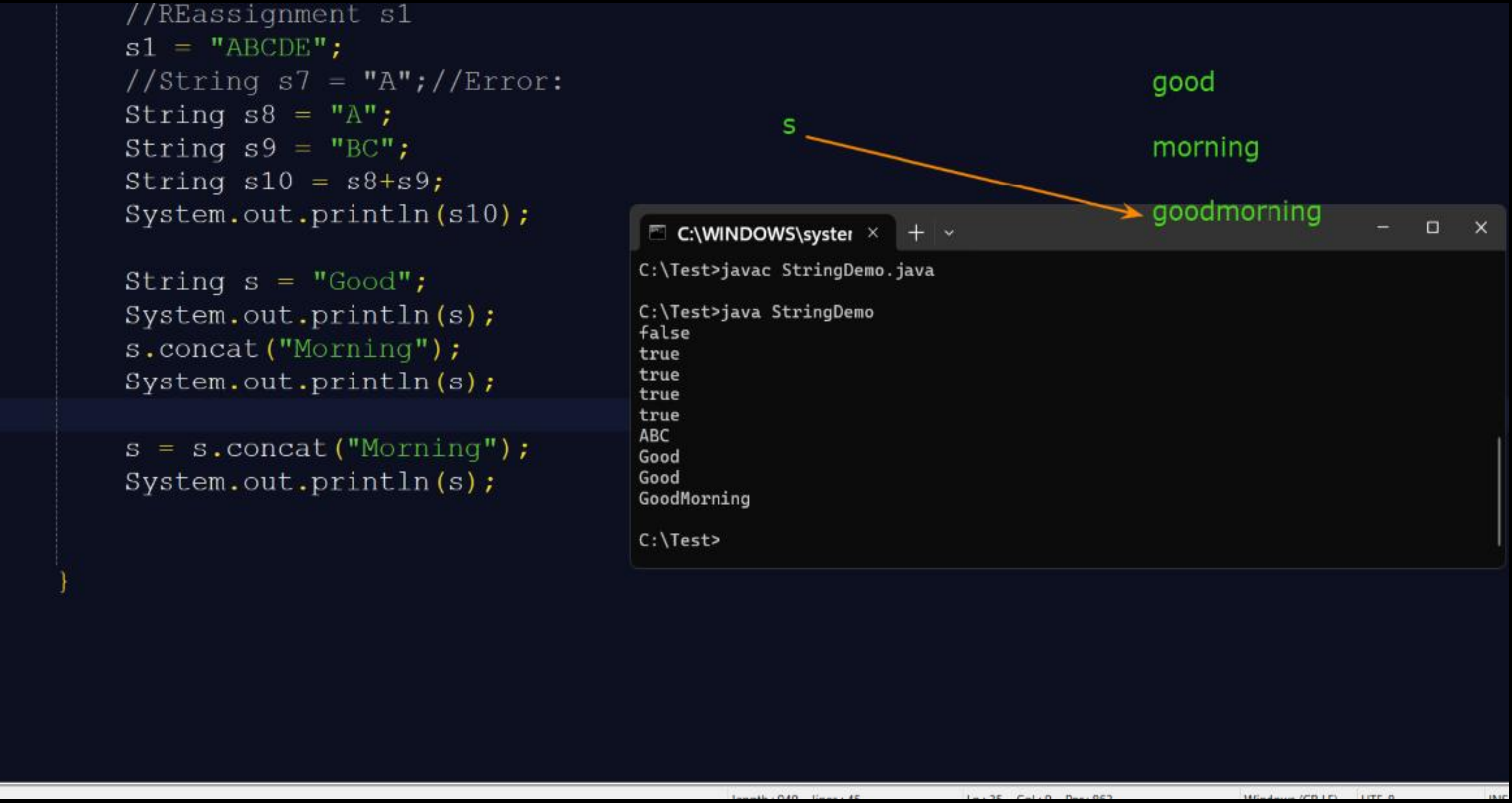


Fig: Hierarchy diagram of Number class in Java


Object Class
getClass()
hashCode()
wait()
toString()
clone()
equals()
finalize()
notify()
notifyall()

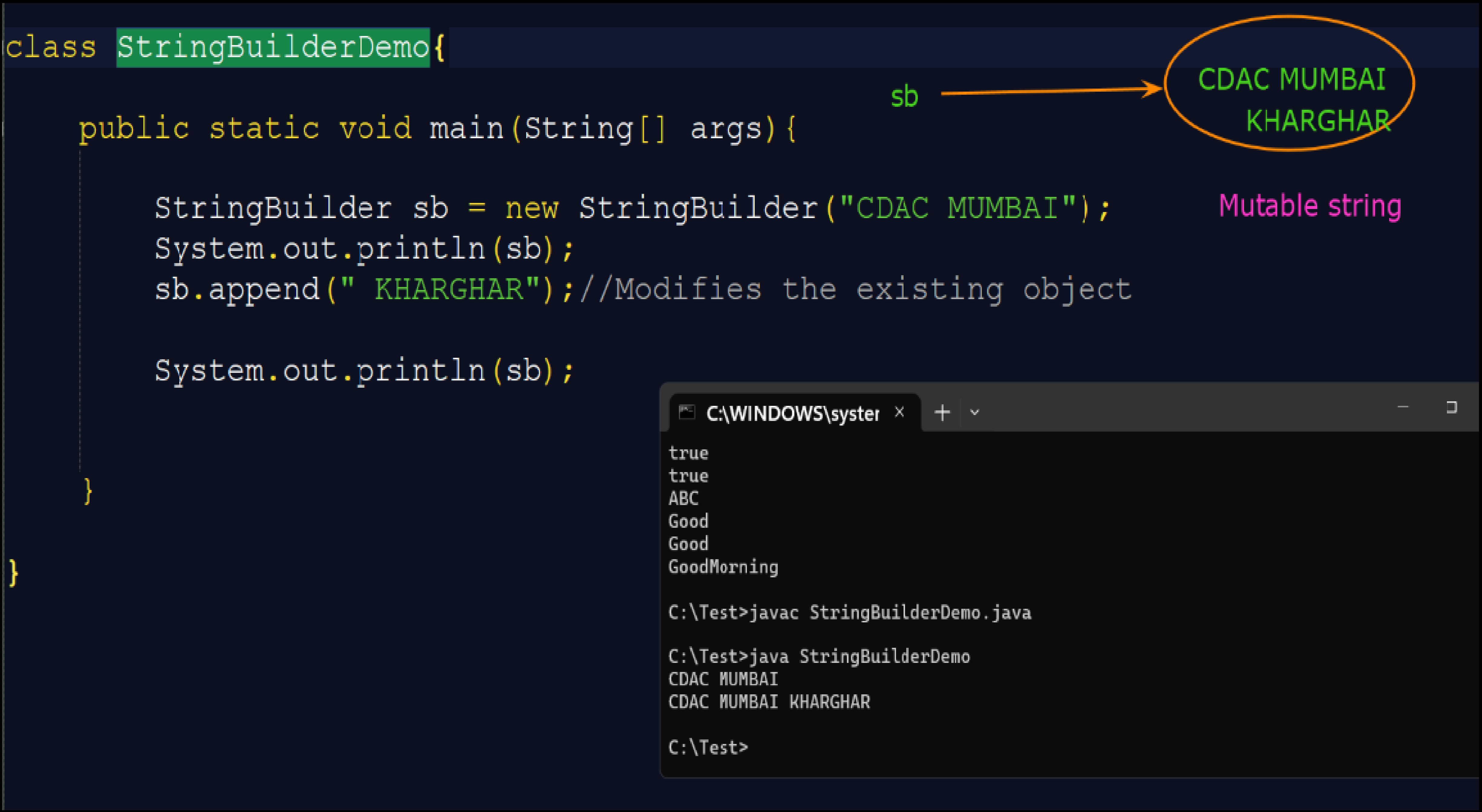


```
class StringDemo{

    public static void main(String... args){
        String s1 = "ABC";//String Literal
        String s2 = "ABCD";
        String s3 = "ABC";
        String s4 = "ABCD";
        String s5 = "ABCDE";

        System.out.println(s1 == s2);//binary path: Object references
        System.out.println(s1 == s3);//binary path: Object references

        String s6 = new String("ABC");
        String s7 = new String("ABCD");
        s5 = new String("ABC");
        s6=s7;
        System.out.println(s1.equals(s3));//Value of reference
        System.out.println(s6.equals(s7));//Value of reference
        System.out.println(s6 == s7);

        String s8 = "Hello";
        String s9 = "Hello";
```



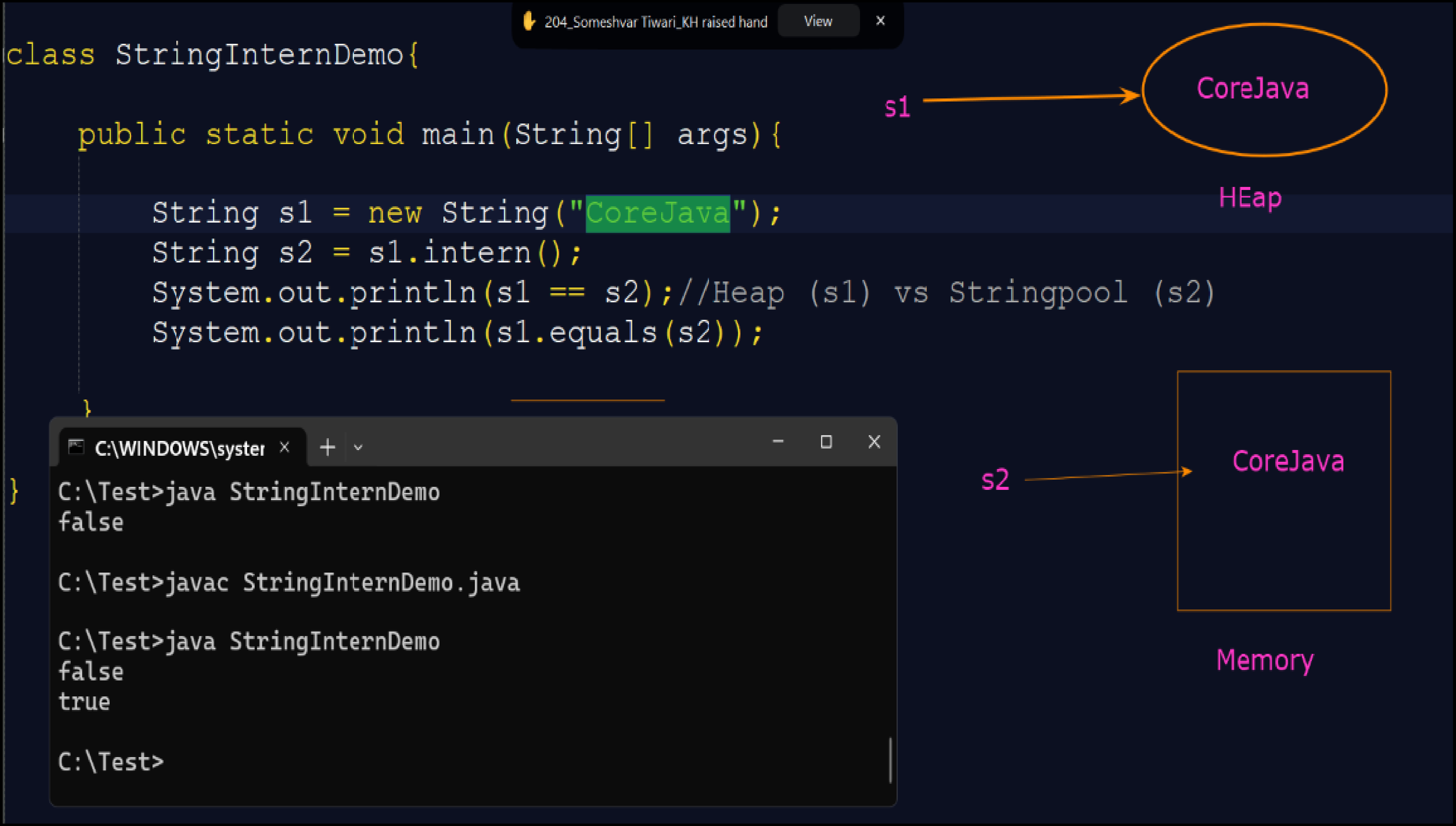
```java
    //REassignment s1
    s1 = "ABCDE";
    //String s7 = "A";//Error:
    String s8 = "A";
    String s9 = "BC";
    String s10 = s8+s9;
    System.out.println(s10);

    String s = "Good";
    System.out.println(s);
    s.concat("Morning");
    System.out.println(s);

    s = s.concat("Morning");
    System.out.println(s);

}
```

```
C:\WINDOWS\syster

C:\Test>javac StringDemo.java

C:\Test>java StringDemo
false
true
true
true
true
ABC
Good
Good
GoodMorning

C:\Test>
```



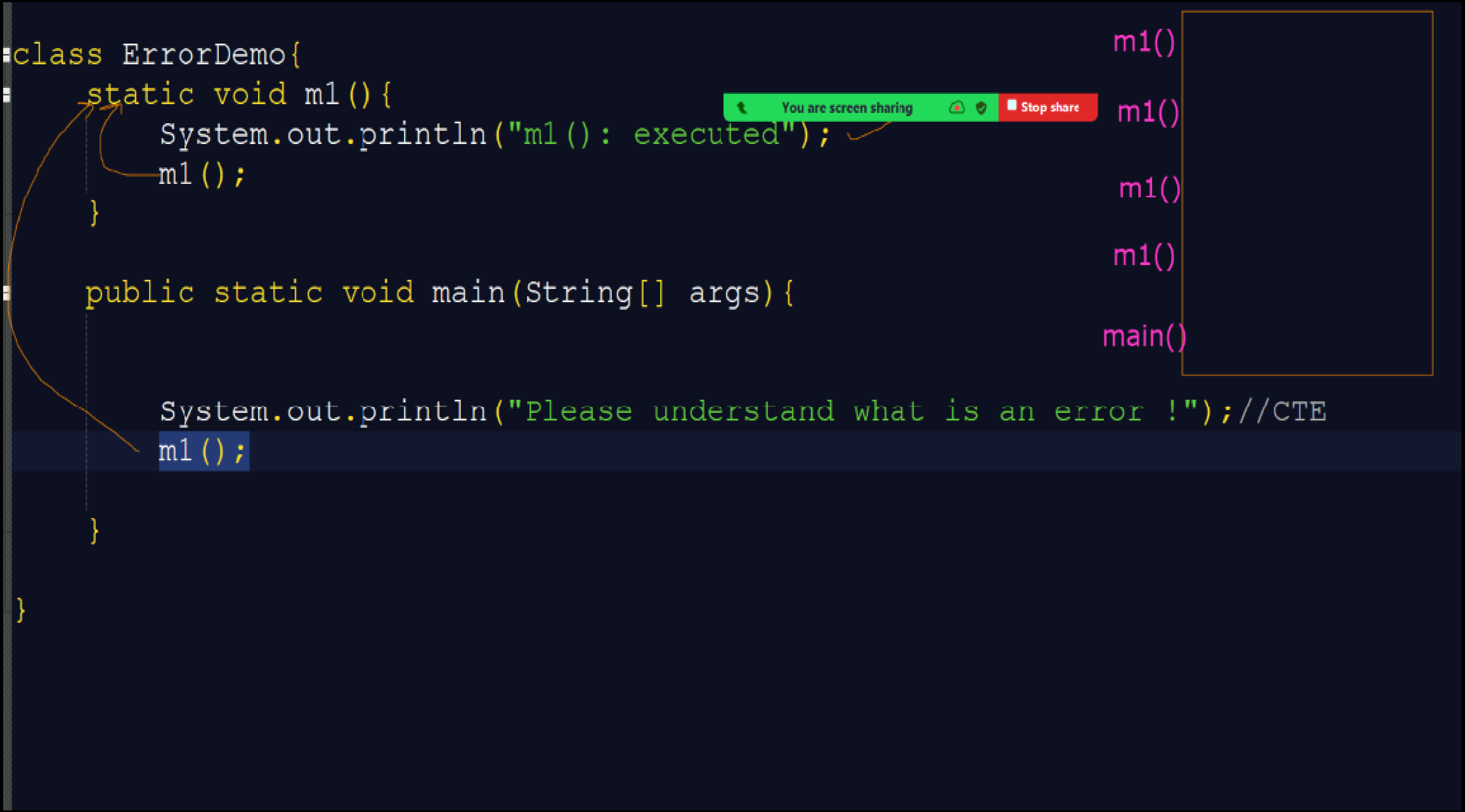
```
class StringBuilderDemo{

    public static void main(String[] args){

        StringBuilder sb = new StringBuilder("CDAC MUMBAI");
        System.out.println(sb);
        sb.append(" KHARGHAR");//Modifies the existing object

        System.out.println(sb);


    }

}
```

```
true
true
ABC
Good
Good
GoodMorning

C:\Test>javac StringBuilderDemo.java

C:\Test>java StringBuilderDemo
CDAC MUMBAI
CDAC MUMBAI KHARGHAR

C:\Test>
```



```java
class StringInternDemo{

    public static void main(String[] args){

        String s1 = new String("CoreJava");
        String s2 = s1.intern();
        System.out.println(s1 == s2);//Heap (s1) vs Stringpool (s2)
        System.out.println(s1.equals(s2));

    }
}
```

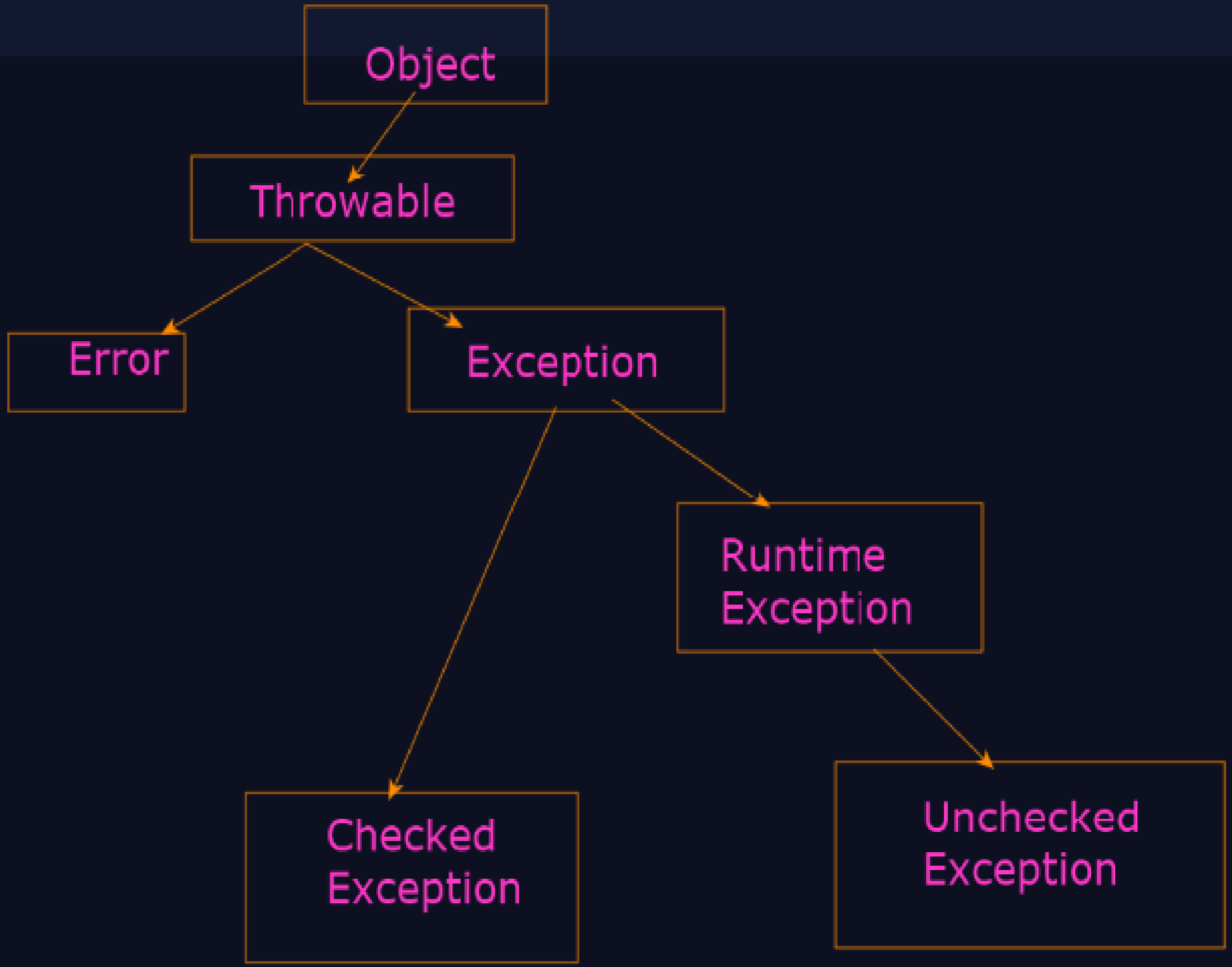
```
C:\Test>java StringInternDemo
false

C:\Test>javac StringInternDemo.java

C:\Test>java StringInternDemo
false
true

C:\Test>
```

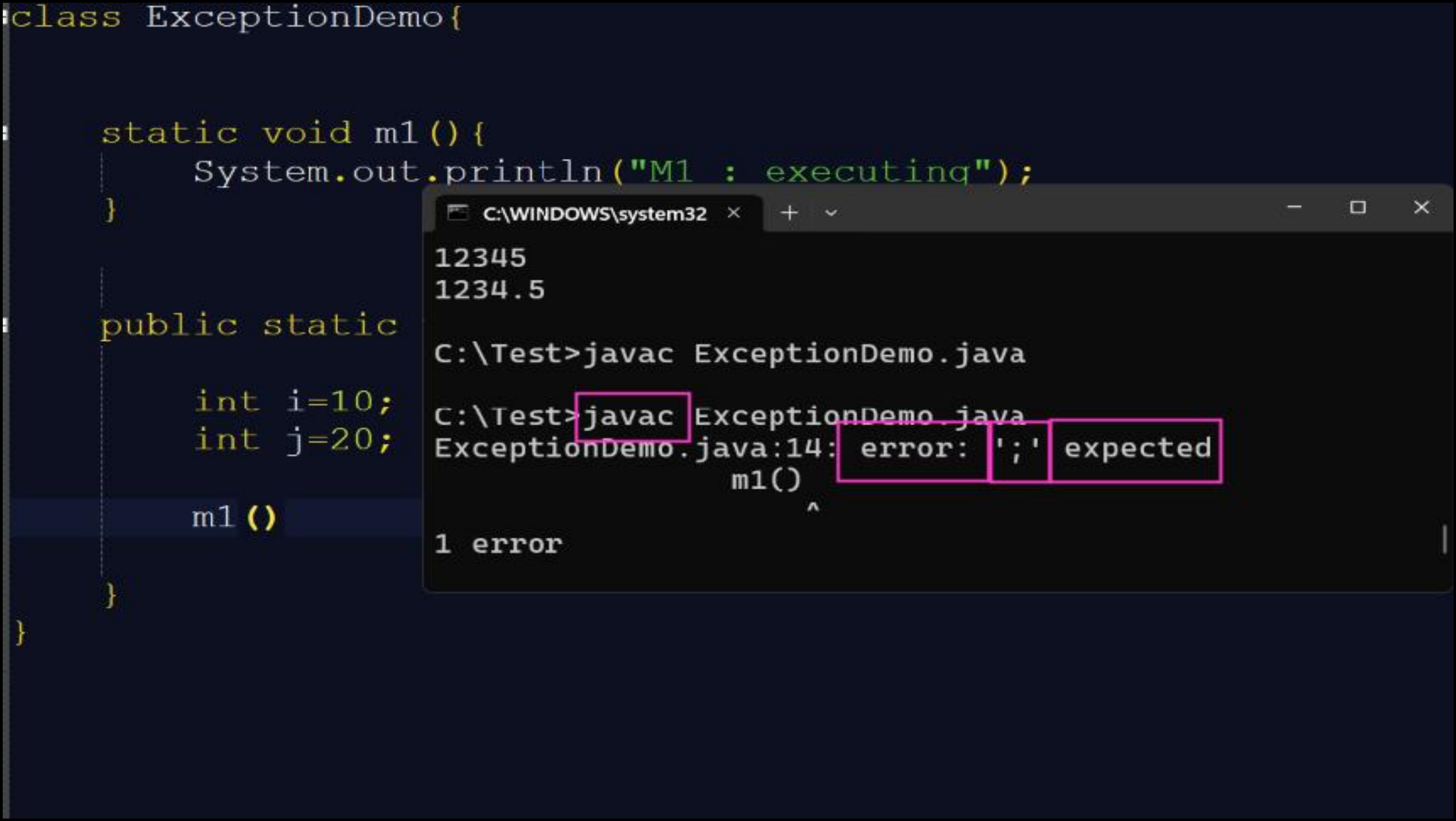


```
class ErrorDemo{
    static void m1(){
        System.out.println("m1(): executed");
        m1();
    }

    public static void main(String[] args){


        System.out.println("Please understand what is an error !");//CTE
        m1();

    }

}
```

-Exception occurs when the normal execution flow is distrupted.

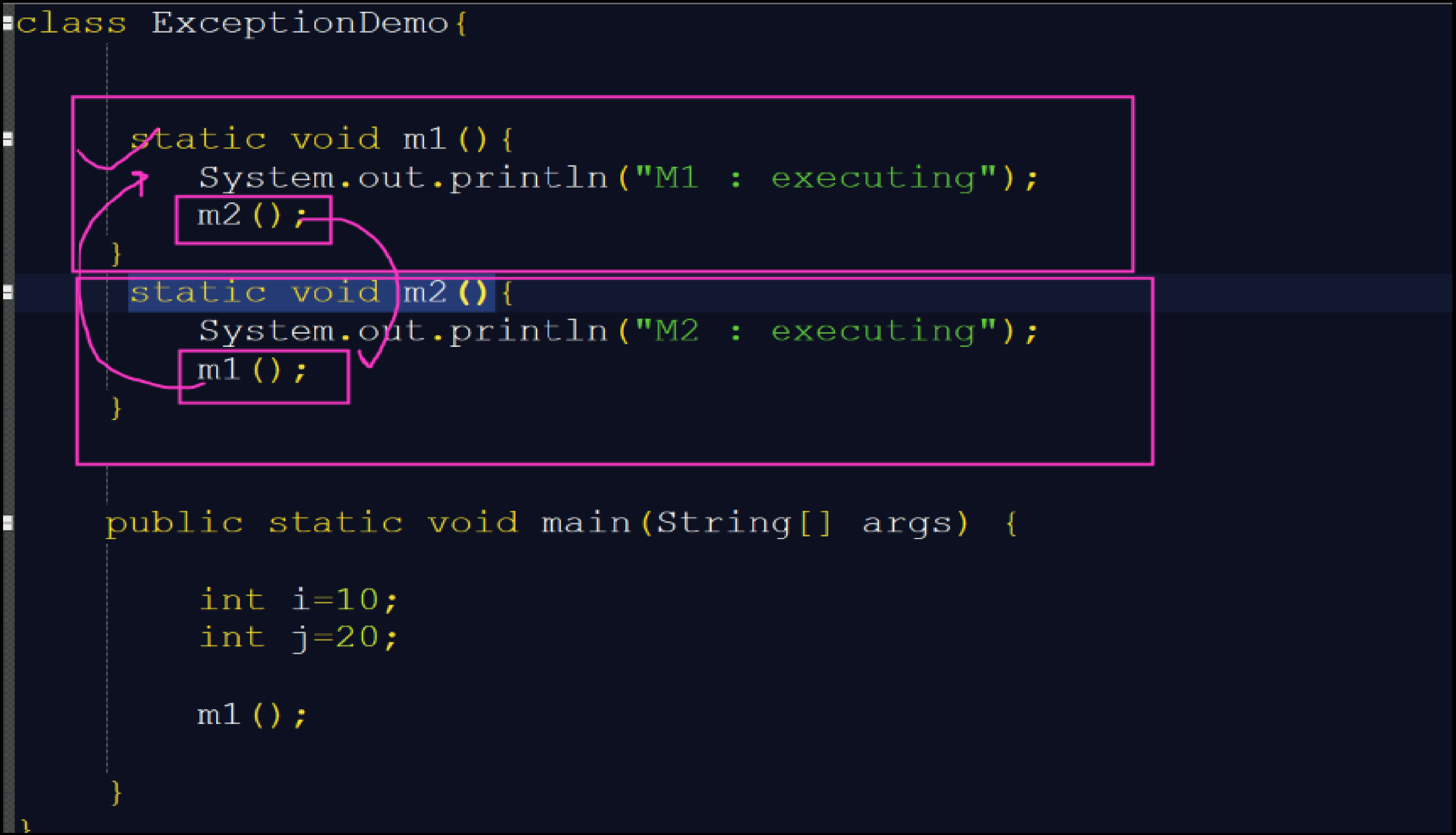
-exception handling : mechanisum :try, catch, finally, throw, throws



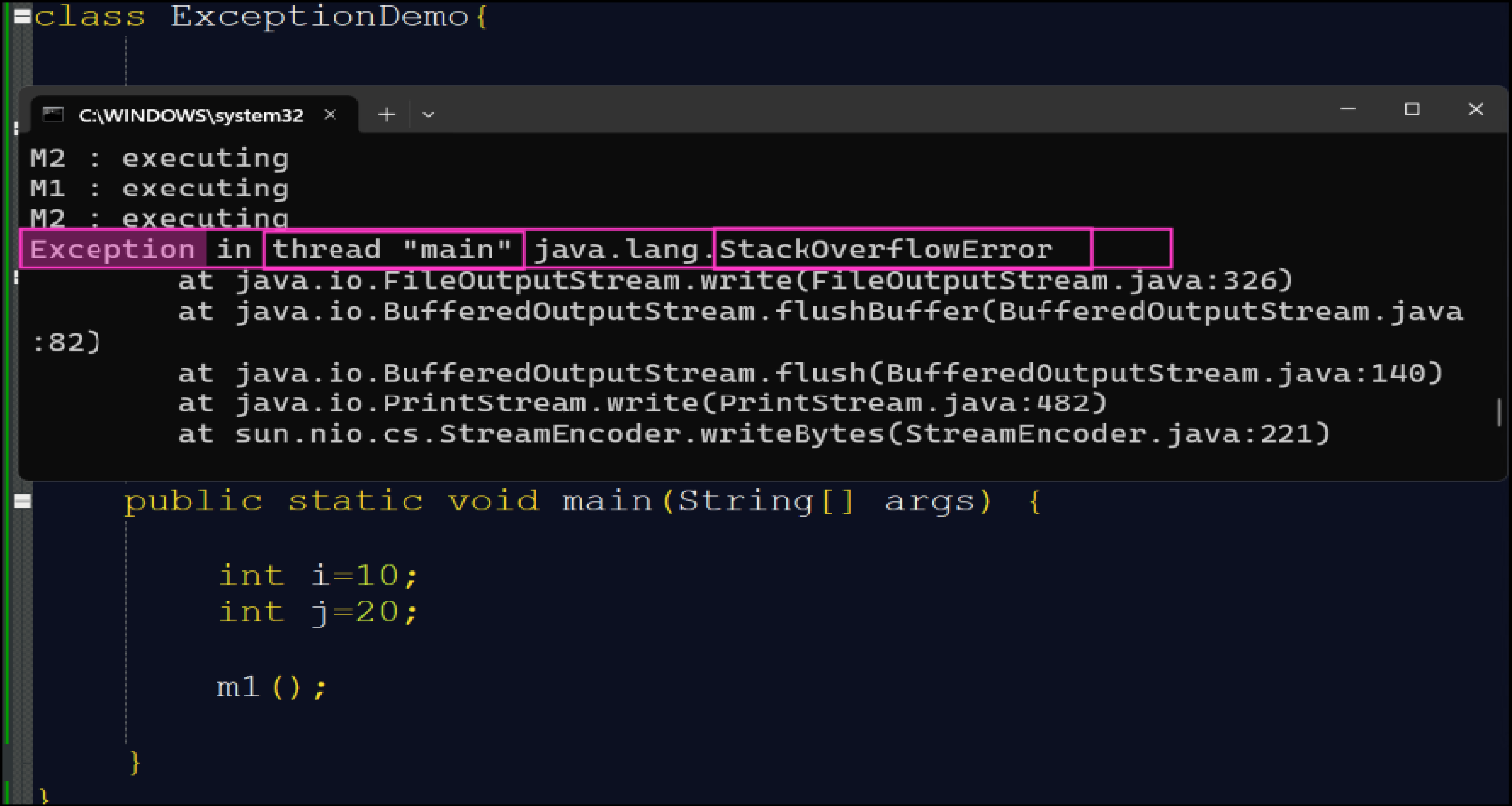


```
class ExceptionDemo{


    static void m1(){
        System.out.println("M1 : executing");
    }


    public static

        int i=10;
        int j=20;

        m1()

    }
}
```

```
C:\WINDOWS\system32

12345
1234.5

C:\Test>javac ExceptionDemo.java

C:\Test>javac ExceptionDemo.java
ExceptionDemo.java:14: error: ';' expected
                m1()
                    ^
1 error
```



```
class ExceptionDemo{


    static void m1(){
        System.out.println("M1 : executing");
        m2();
    }
    static void m2(){
        System.out.println("M2 : executing");
        m1();
    }


    public static void main(String[] args) {

        int i=10;
        int j=20;

        m1();

    }
}
```



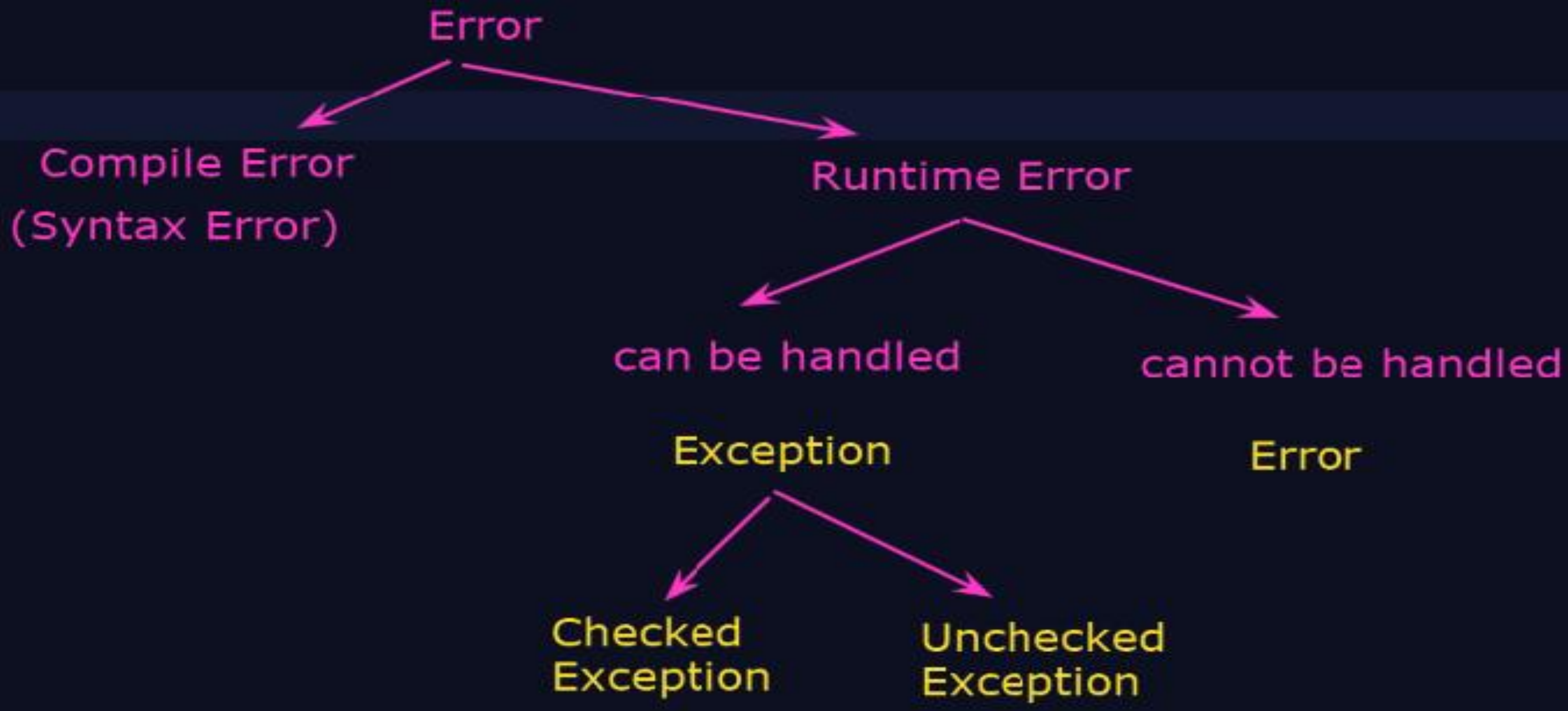
```
class ExceptionDemo{
```

C:\WINDOWS\system32

```
M2 : executing
M1 : executing
M2 : executing
Exception in thread "main" java.lang.StackOverflowError
        at java.io.FileOutputStream.write(FileOutputStream.java:326)
        at java.io.BufferedOutputStream.flushBuffer(BufferedOutputStream.java
:82)
        at java.io.BufferedOutputStream.flush(BufferedOutputStream.java:140)
        at java.io.PrintStream.write(PrintStream.java:482)
        at sun.nio.cs.StreamEncoder.writeBytes(StreamEncoder.java:221)
```

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

        int i=10;
        int j=20;

        m1();

    }
}
```

Exception Handling:



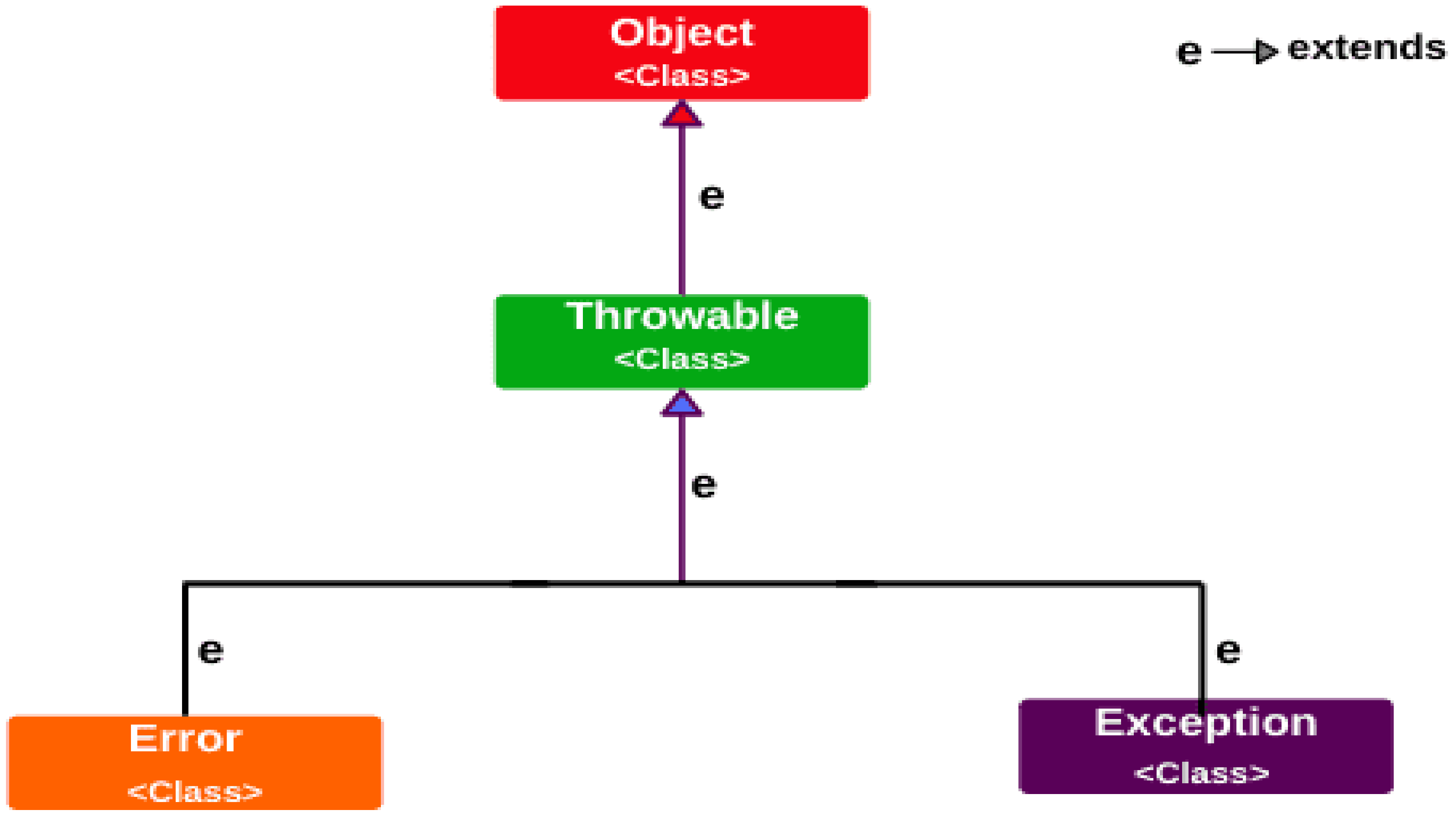


Fig: Java Exception Hierarchy

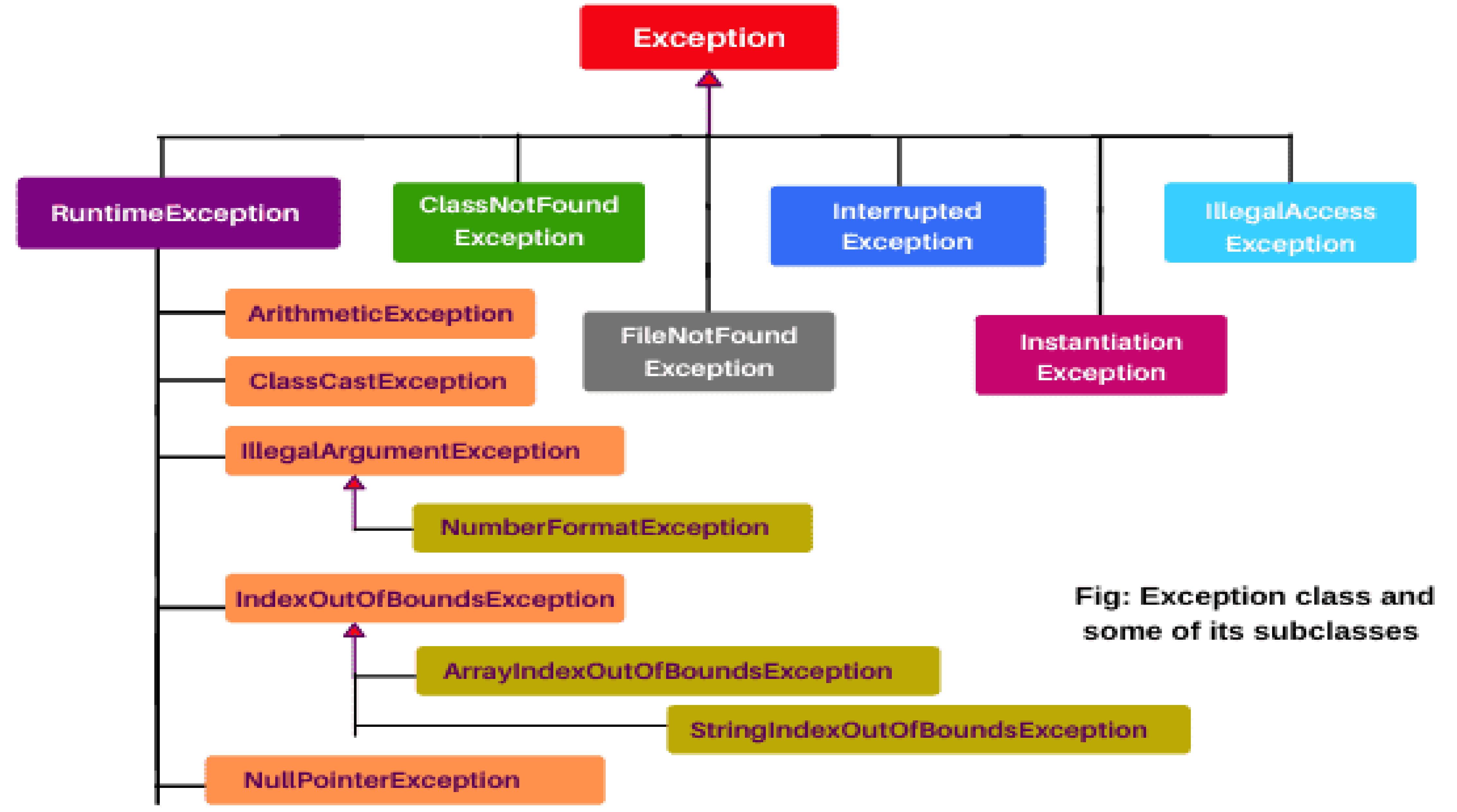
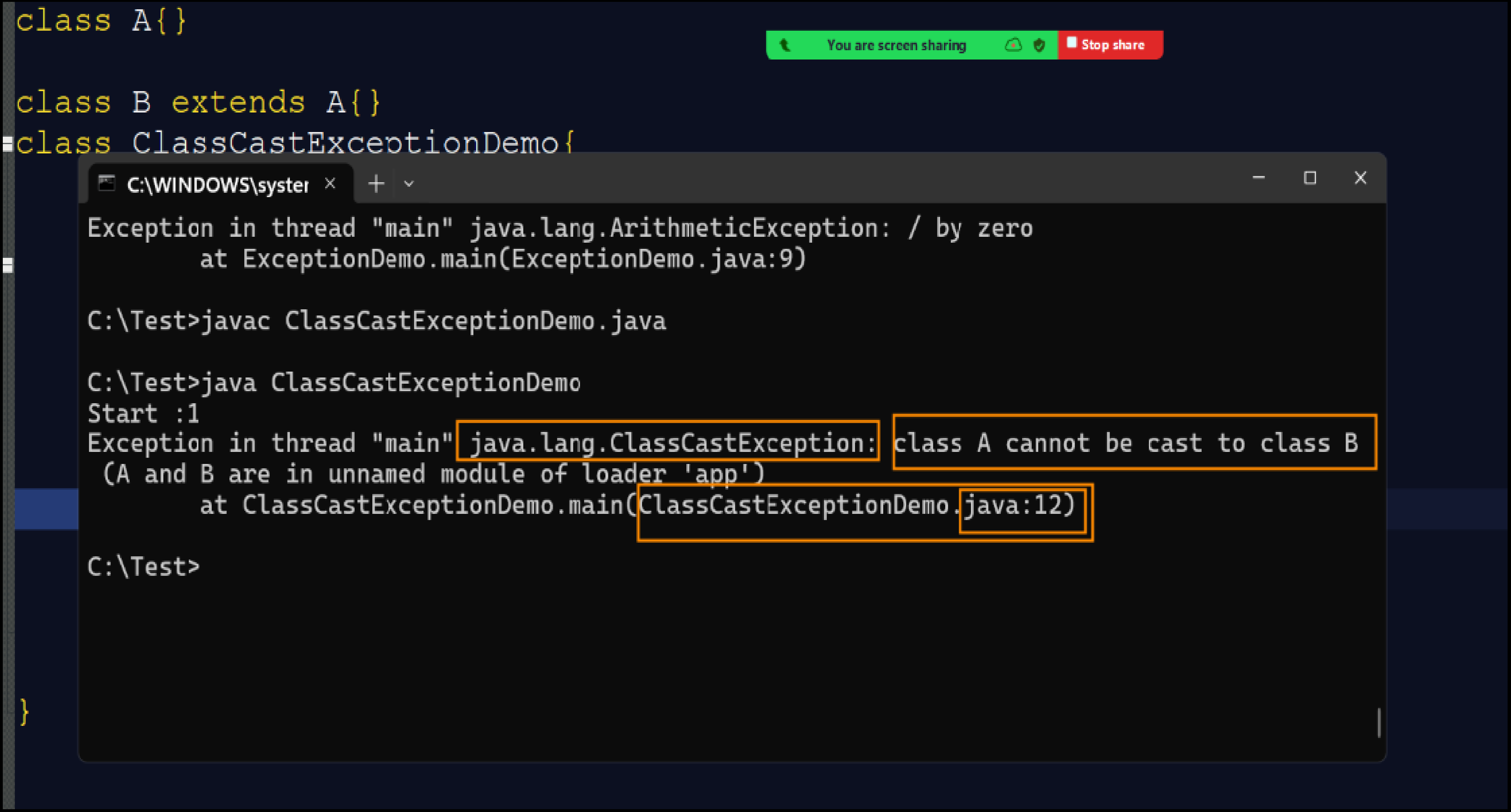


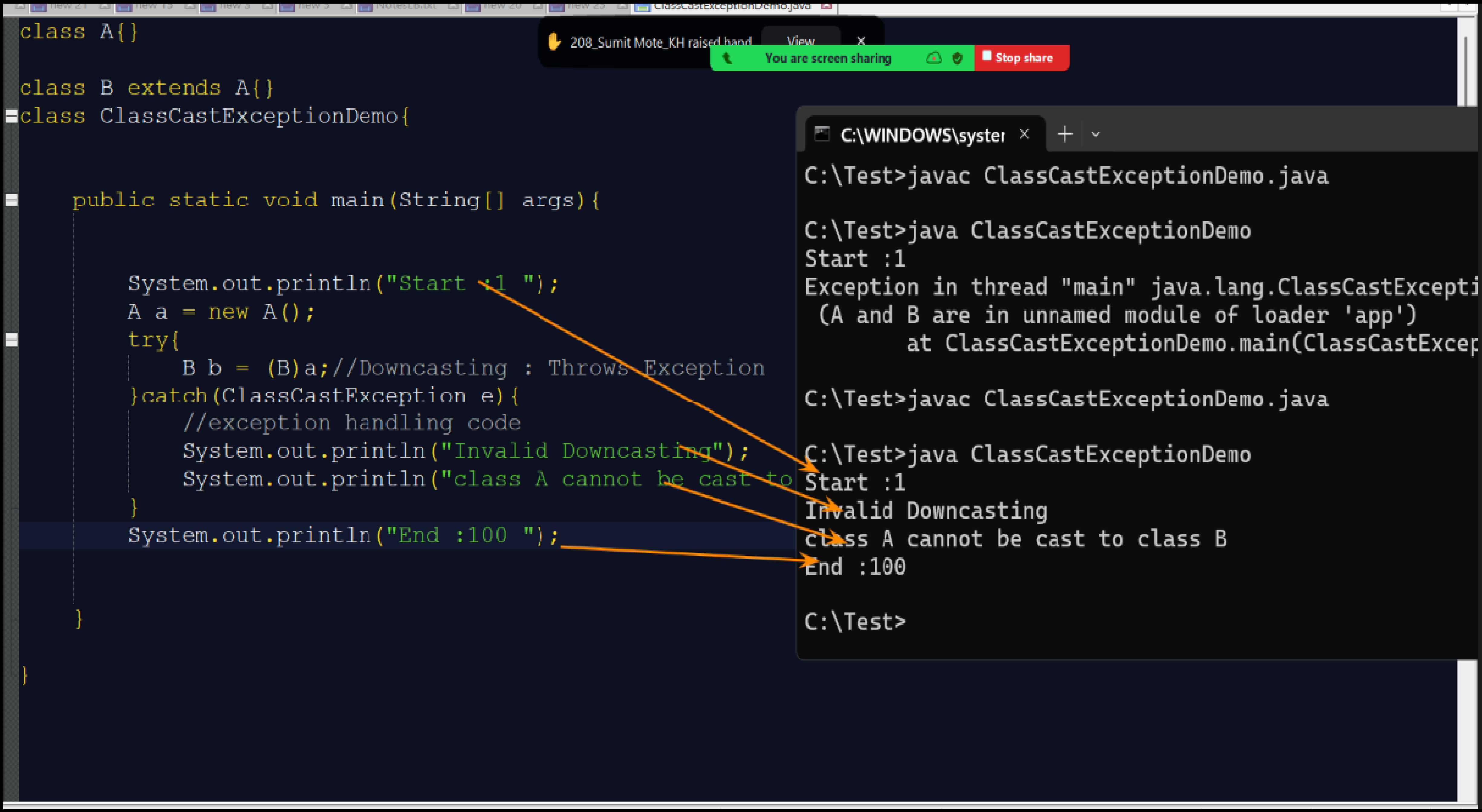
Fig: Exception class and some of its subclasses



```
class A{}

class B extends A{}
class ClassCastExceptionDemo{
```

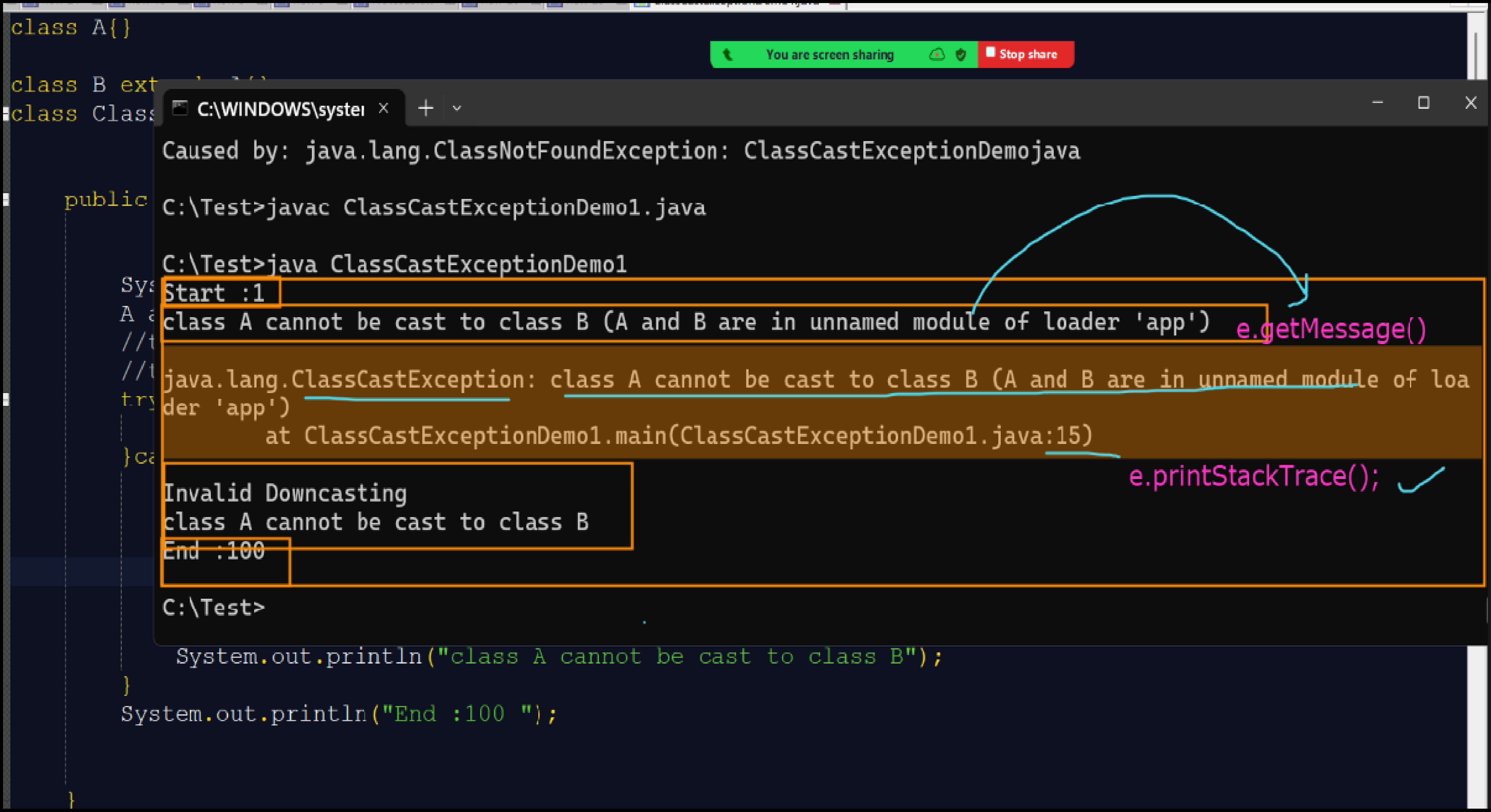
```
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at ExceptionDemo.main(ExceptionDemo.java:9)

C:\Test>javac ClassCastExceptionDemo.java

C:\Test>java ClassCastExceptionDemo
Start :1
Exception in thread "main" java.lang.ClassCastException: class A cannot be cast to class B
 (A and B are in unnamed module of loader 'app')
        at ClassCastExceptionDemo.main(ClassCastExceptionDemo.java:12)

C:\Test>
```

```
}
```



```
class A{}

class B extends A{}
class ClassCastExceptionDemo{


    public static void main(String[] args){


        System.out.println("Start :1 ");
        A a = new A();
        try{
            B b = (B)a;//Downcasting : Throws Exception
        }catch(ClassCastException e){
            //exception handling code
            System.out.println("Invalid Downcasting");
            System.out.println("class A cannot be cast to
        }
        System.out.println("End :100 ");


    }

}
```

```
C:\Test>javac ClassCastExceptionDemo.java

C:\Test>java ClassCastExceptionDemo
Start :1
Exception in thread "main" java.lang.ClassCastExcepti
 (A and B are in unnamed module of loader 'app')
        at ClassCastExceptionDemo.main(ClassCastExcep

C:\Test>javac ClassCastExceptionDemo.java

C:\Test>java ClassCastExceptionDemo
Start :1
Invalid Downcasting
class A cannot be cast to class B
End :100

C:\Test>
```



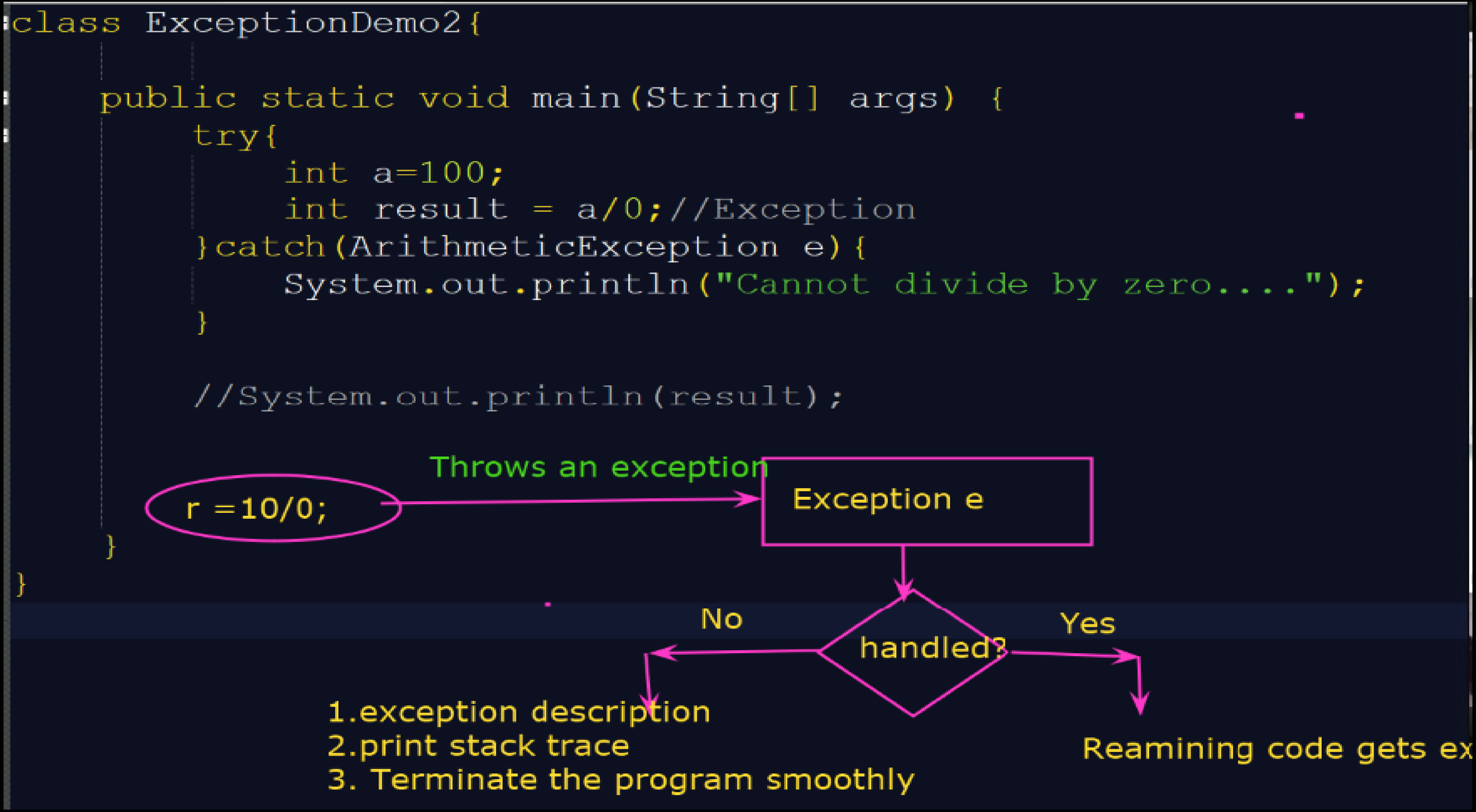
```
class A{}

class B ext
class Class

    public

        Sys
        A a
        //t
        //t
        try

        }ca

            System.out.println("class A cannot be cast to class B");
        }
        System.out.println("End :100 ");

    }
```

```
Caused by: java.lang.ClassNotFoundException: ClassCastExceptionDemojava

C:\Test>javac ClassCastExceptionDemo1.java

C:\Test>java ClassCastExceptionDemo1
Start :1
class A cannot be cast to class B (A and B are in unnamed module of loader 'app')

java.lang.ClassCastException: class A cannot be cast to class B (A and B are in unnamed module of loa
der 'app')
        at ClassCastExceptionDemo1.main(ClassCastExceptionDemo1.java:15)

Invalid Downcasting
class A cannot be cast to class B
End :100

C:\Test>
```

e.getMessage()

e.printStackTrace(); ✓

```
try
{
  // A block of code; // generates an exception
}
catch(exception_class var)
{
  // Code to be executed when an exception is
thrown.
}
```



```
class ExceptionDemo2{

    public static void main(String[] args) {
        try{
            int a=100;
            int result = a/0;//Exception
        }catch(ArithmeticException e){
            System.out.println("Cannot divide by zero....");
        }

        //System.out.println(result);


        r =10/0;
    }
}
```

1.exception description
2.print stack trace
3. Terminate the program smoothly

Reamining code gets ex

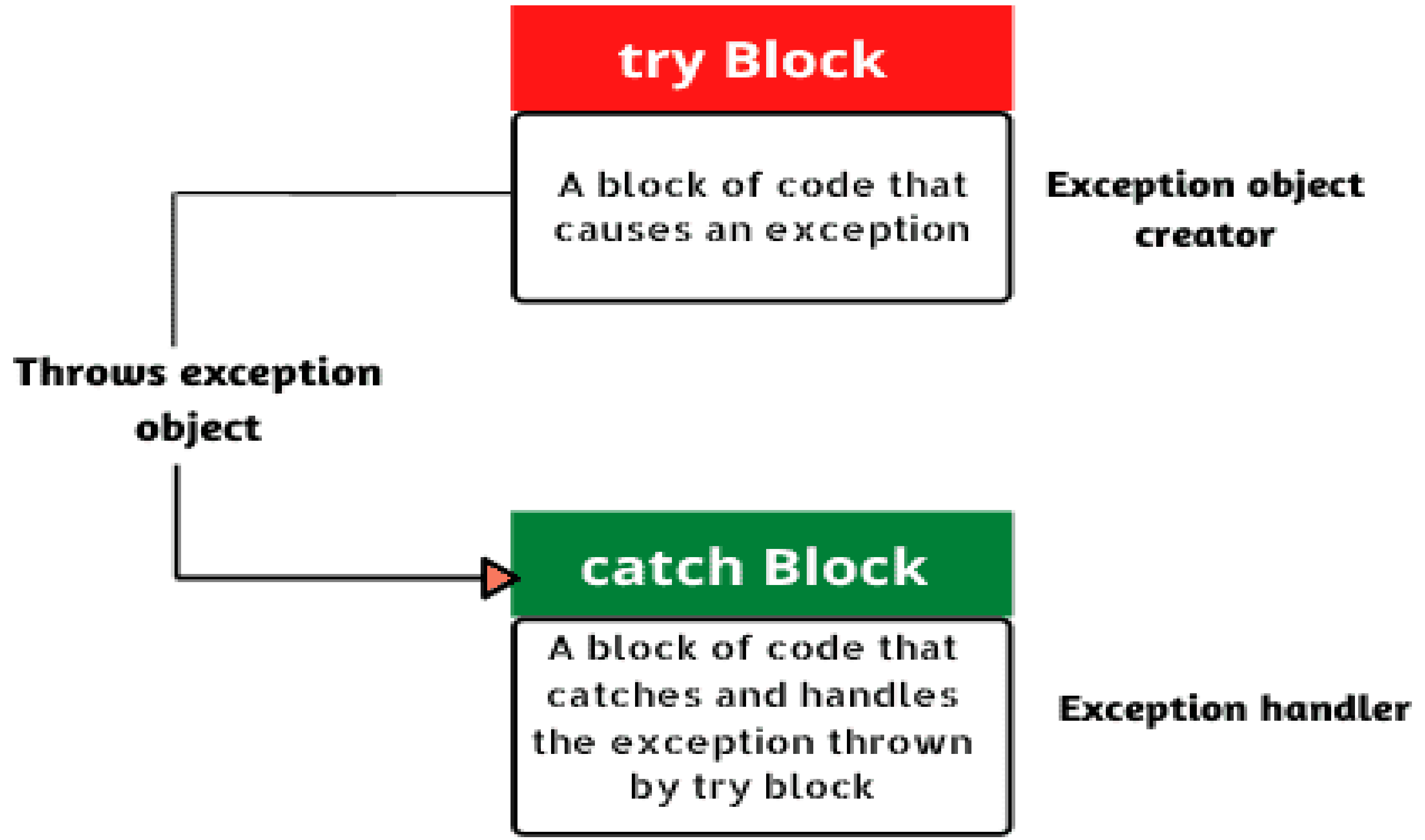
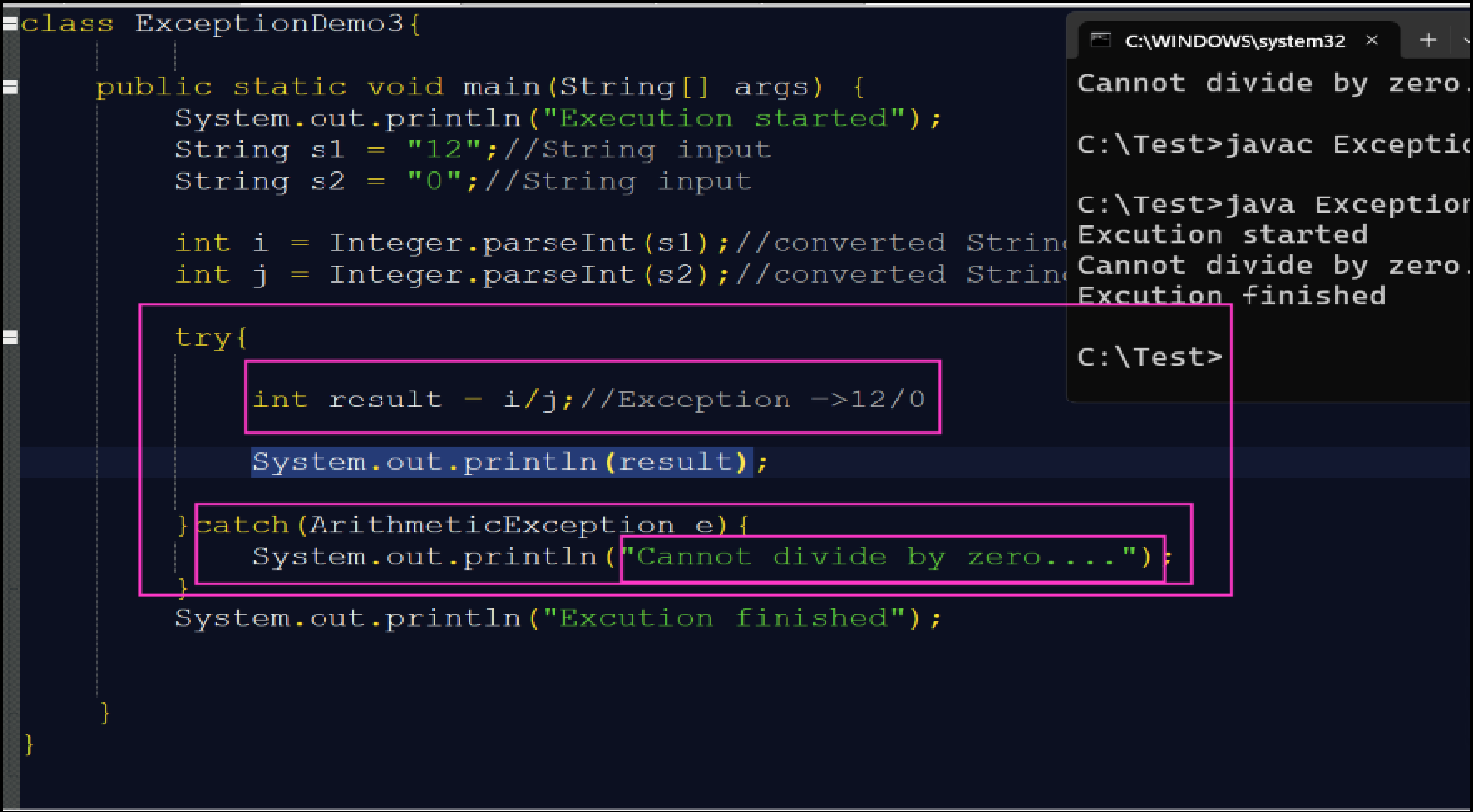

try Block
A block of code that causes an exception
Exception object creator
Throws exception object
catch Block
A block of code that catches and handles the exception thrown by try block
Exception handler

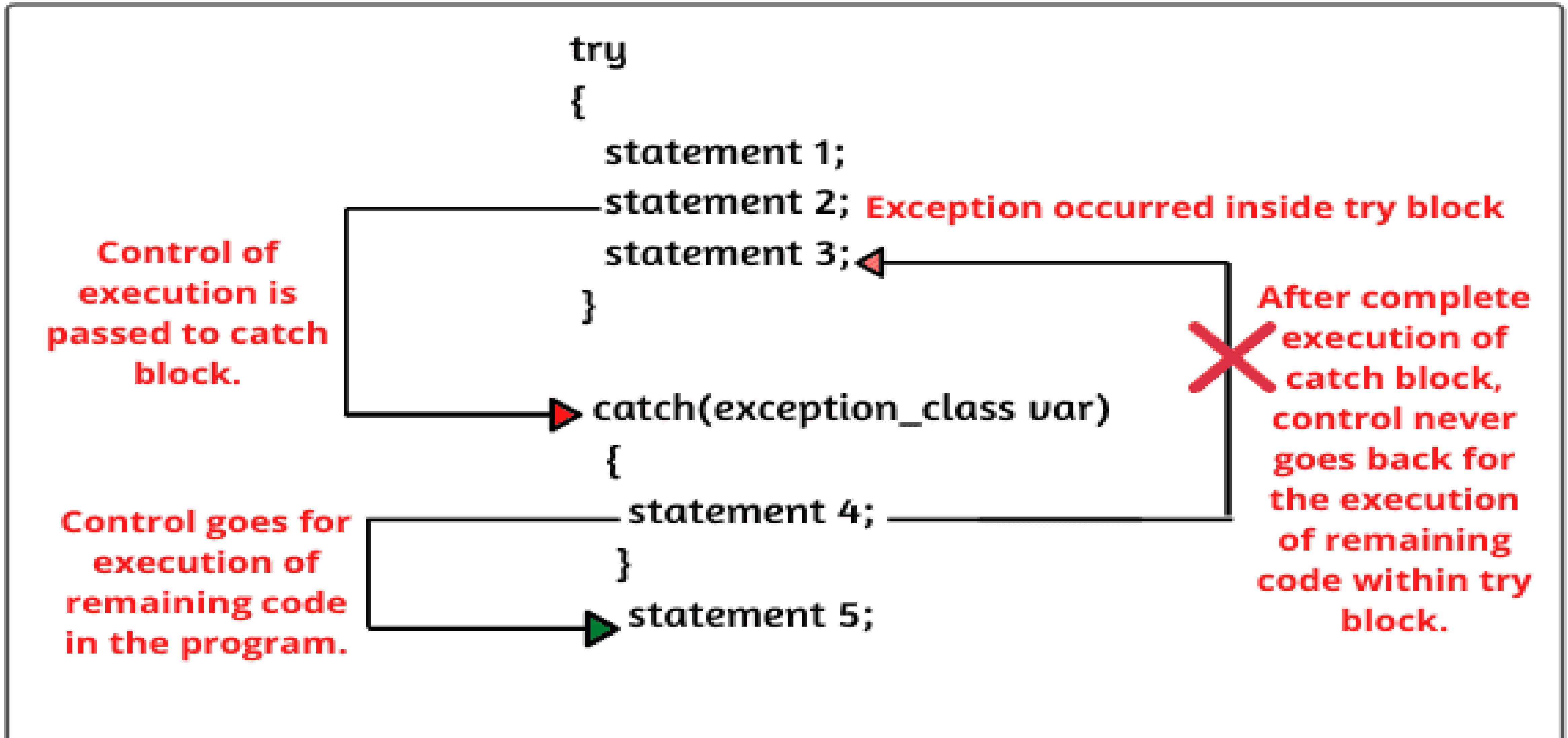
Fig: Exception handling mechanism



```
class ExceptionDemo3{

    public static void main(String[] args) {
        System.out.println("Execution started");
        String s1 = "12";//String input
        String s2 = "0";//String input

        int i = Integer.parseInt(s1);//converted Strin
        int j = Integer.parseInt(s2);//converted Strin

        try{

            int result = i/j;//Exception ->12/0

            System.out.println(result);

        }catch(ArithmeticException e){
            System.out.println("Cannot divide by zero....");
        }
        System.out.println("Excution finished");


    }
}
```

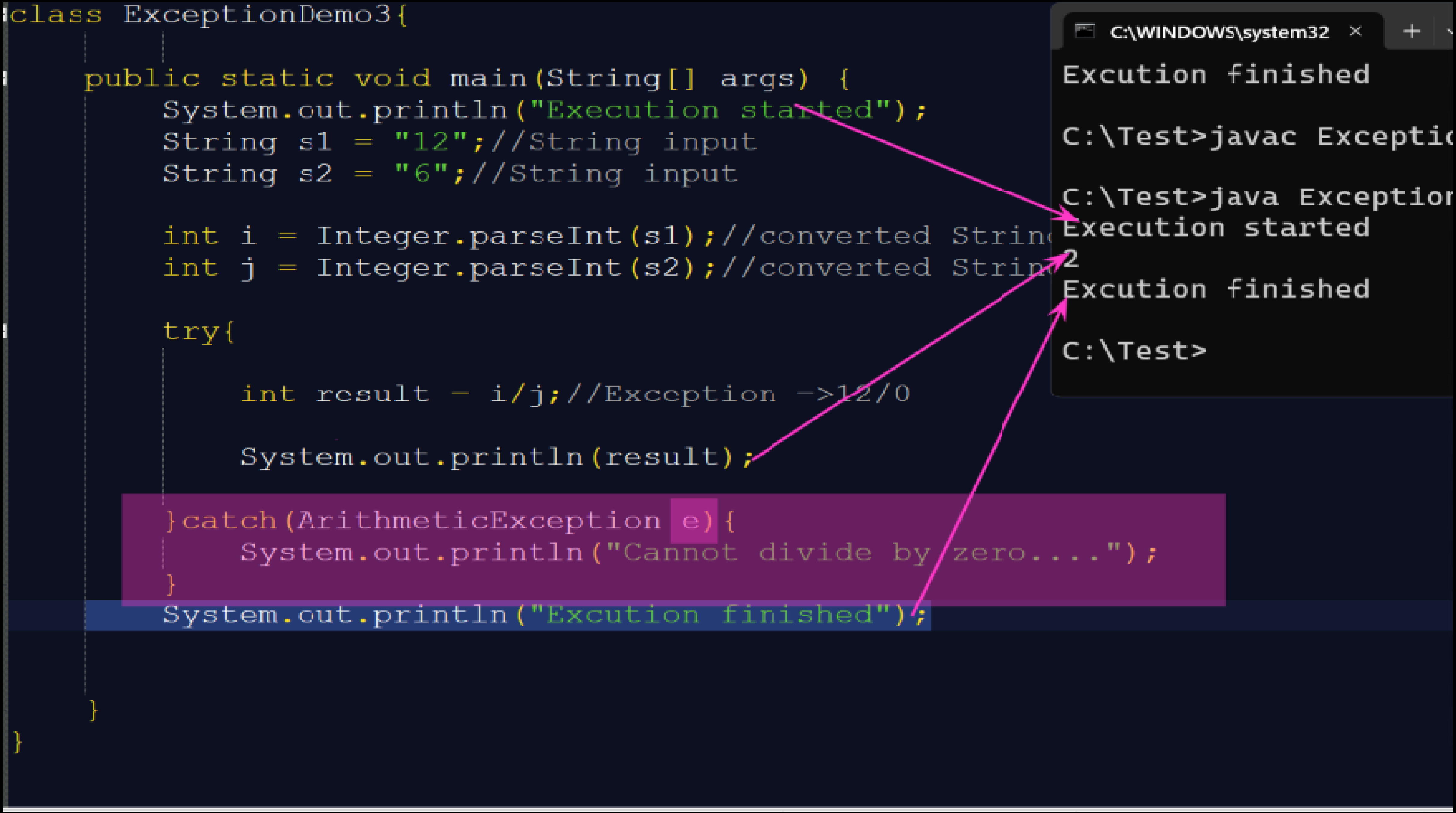
C:\WINDOWS\system32

```
Cannot divide by zero.

C:\Test>javac Exceptio

C:\Test>java Exceptior
Excution started
Cannot divide by zero.
Excution finished

C:\Test>
```



try
{
statement 1;
statement 2;
statement 3;
}
catch(exception_class var)
{
statement 4;
}
statement 5;
Exception occurred inside try block
Control of execution is passed to catch block.
After complete execution of catch block, control never goes back for the execution of remaining code within try block.
Control goes for execution of remaining code in the program.

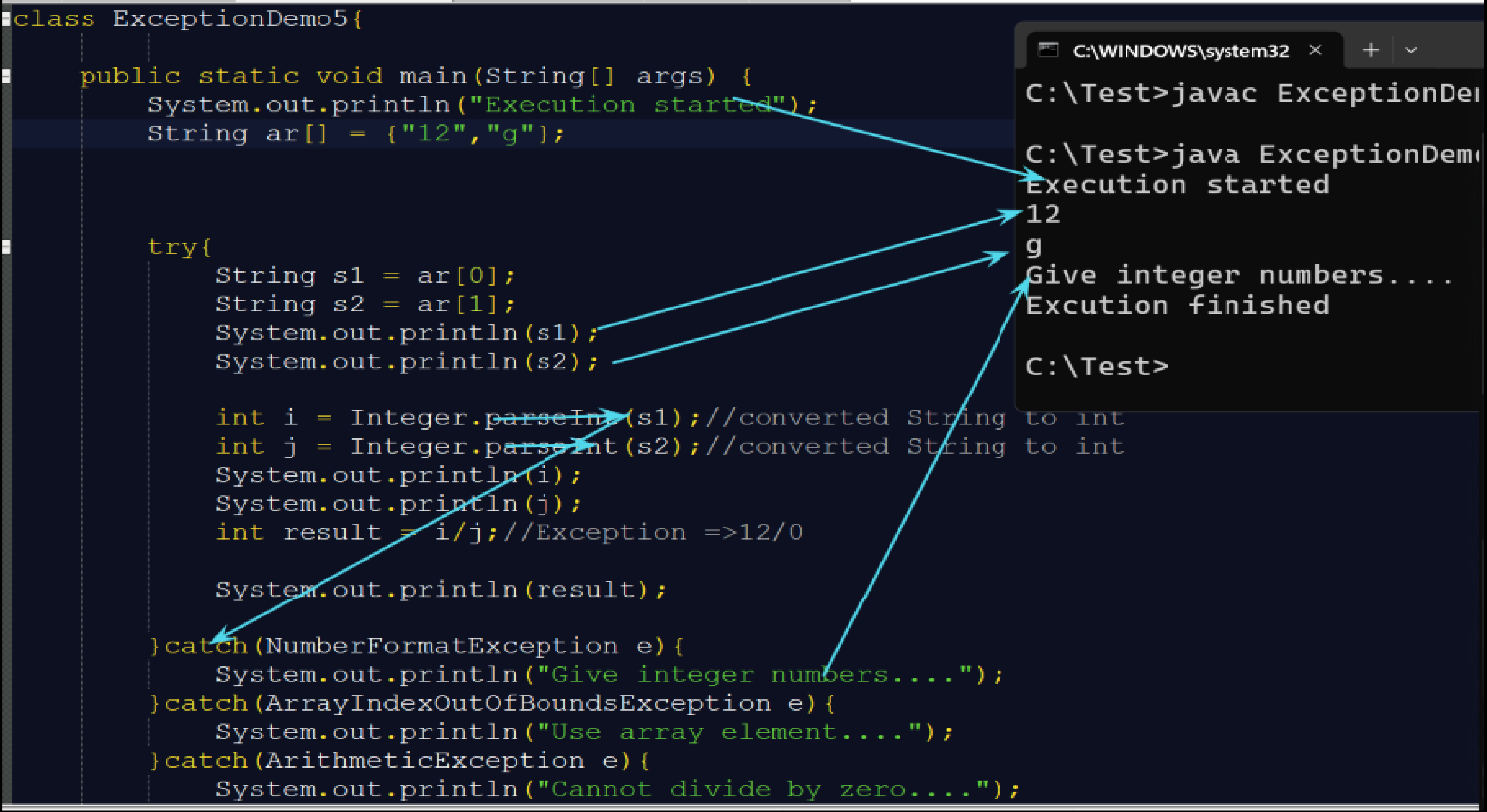
Fig: Control flow of try-catch block in Java



```java
class ExceptionDemo3{

    public static void main(String[] args) {
        System.out.println("Execution started");
        String s1 = "12";//String input
        String s2 = "6";//String input

        int i = Integer.parseInt(s1);//converted Strin
        int j = Integer.parseInt(s2);//converted Strin

        try{

            int result = i/j;//Exception ->12/0

            System.out.println(result);

        }catch(ArithmeticException e){
            System.out.println("Cannot divide by zero....");
        }
        System.out.println("Excution finished");


    }
}
```

C:\WINDOWS\system32

```
Excution finished

C:\Test>javac Exceptic

C:\Test>java Exceptior
Execution started
2
Excution finished

C:\Test>
```



```
class ExceptionDemo5{

	public static void main(String[] args) {
		System.out.println("Execution started");
		String ar[] = {"12","g"};



		try{
			String s1 = ar[0];
			String s2 = ar[1];
			System.out.println(s1);
			System.out.println(s2);

			int i = Integer.parseInt(s1);//converted String to int
			int j = Integer.parseInt(s2);//converted String to int
			System.out.println(i);
			System.out.println(j);
			int result = i/j;//Exception =>12/0

			System.out.println(result);

		}catch(NumberFormatException e){
			System.out.println("Give integer numbers....");
		}catch(ArrayIndexOutOfBoundsException e){
			System.out.println("Use array element....");
		}catch(ArithmeticException e){
			System.out.println("Cannot divide by zero....");
```

C:\WINDOWS\system32

```
C:\Test>javac ExceptionDem

C:\Test>java ExceptionDem
Execution started
12
g
Give integer numbers....
Excution finished

C:\Test>
```