Object Oriented Programming with Java (OOPJ)

Session 5: Arrays

Kiran Waghmare

```java
class GarbageCollectionDemo{
    public static void main(String args[]){//method


    }


}
```

C:\WINDOWS\system32

```
ption;
  public java.lang.String toString();
  public final native void notify();
  public final native void notifyAll();
  public final native void wait(long) throws java.lang.InterruptedE
  public final void wait(long, int) throws java.lang.InterruptedExc
  public final void wait() throws java.lang.InterruptedException;
  protected void finalize() throws java.lang.Throwable;
  static {};
}
```

```java
class Employee{
    String name;
    Employee(String name){
        this.name = name;
    }
}
class GarbageCollectionDemo{

    protected void finalize(){//overriding of method
        System.out.println("Finalize method called ....");
    }
    public static void main(String args[]){

        Employee g1 = new Employee("Java");
        g1 = null;

        System.gc();//Request GC
        Runtime.getRuntime().gc();


    }
```

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:\WINDOWS\system32
C:\Test>javac StringDemo.java

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

C:\Test>
```

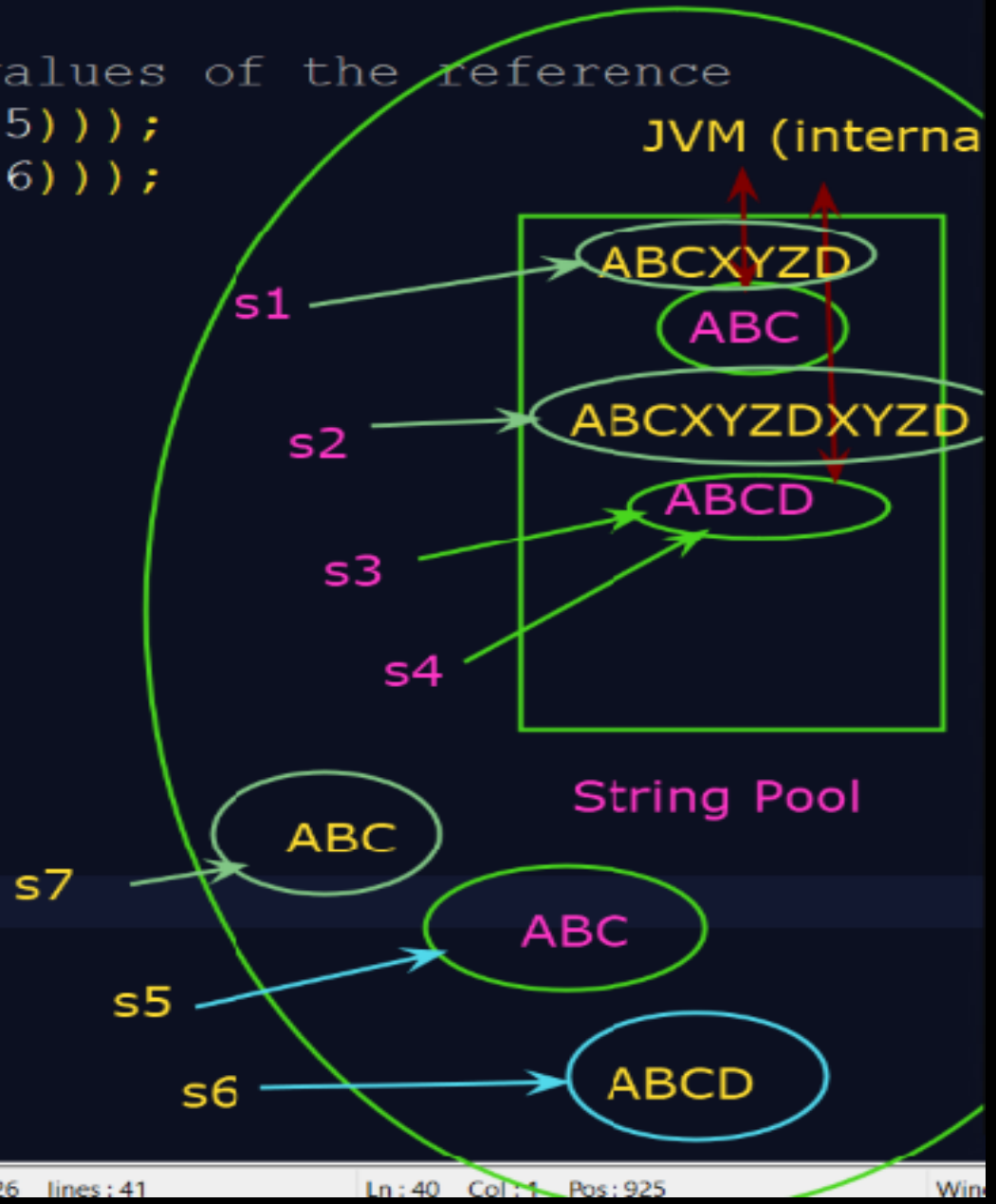
length : 608 lines : 29 Ln : 23 Col : 42 Pos : 590 Win

```java
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");
}
```

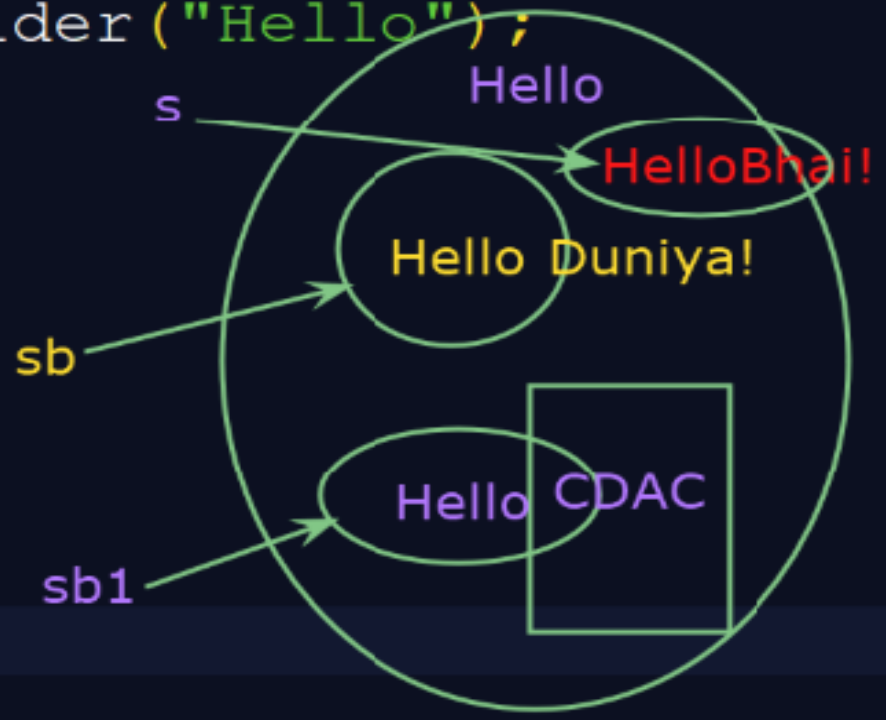


```java
public static void main(String args[]){

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);
}
```



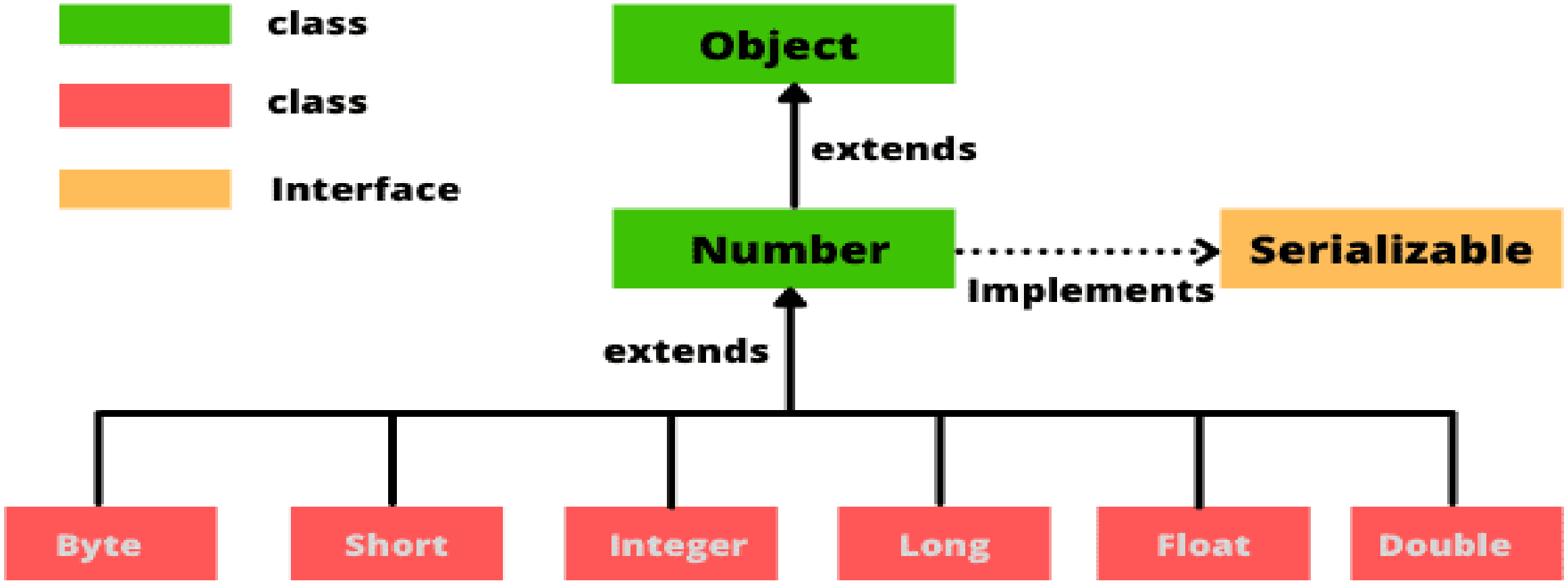
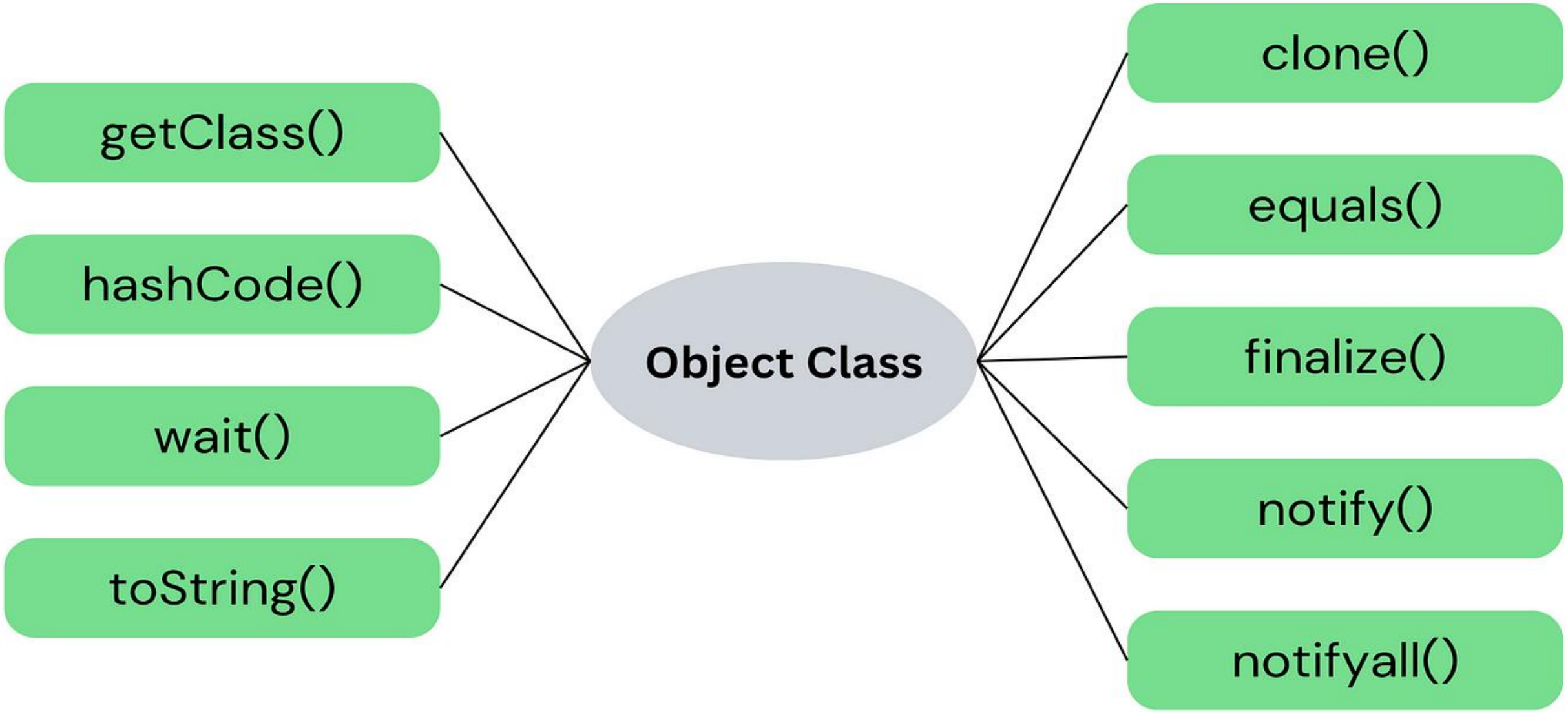


Fig: Hierarchy diagram of Number class in Java


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

```java
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
```

```java
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();

    }
}
```

```java
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)
```

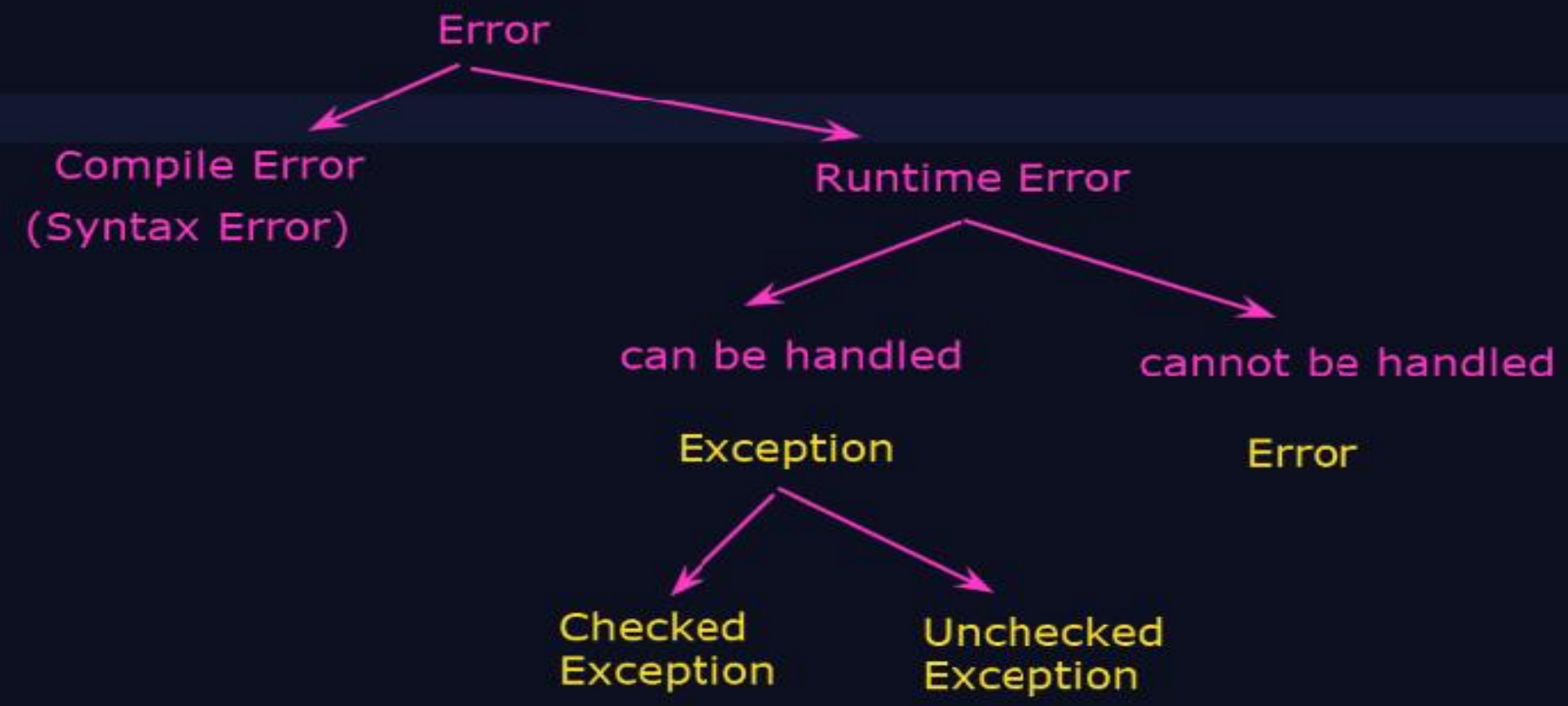
```java
    public static void main(String[] args) {

        int i=10;
        int j=20;

        m1();

    }
}
```

Exception Handling:



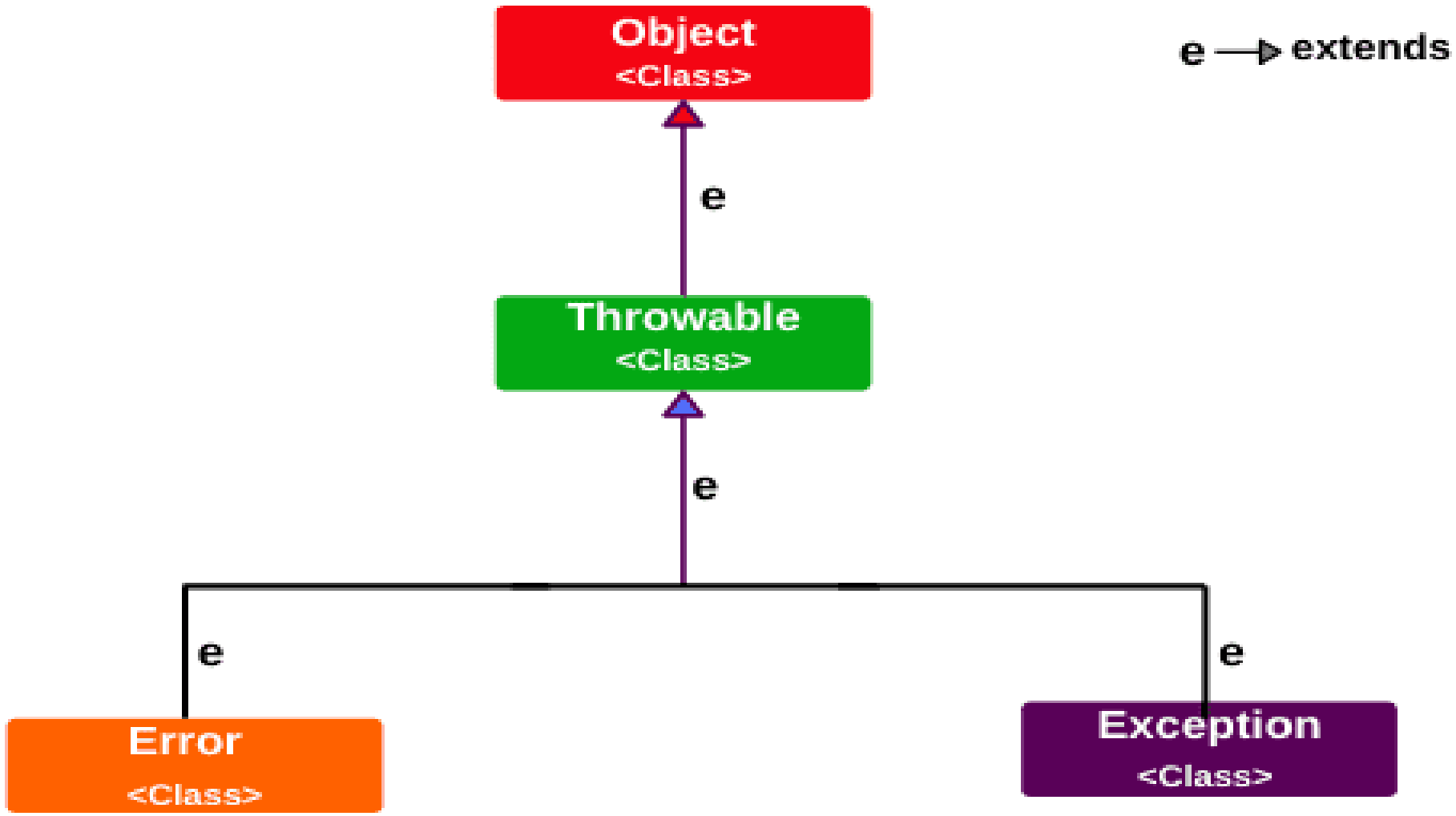


Fig: Java Exception Hierarchy

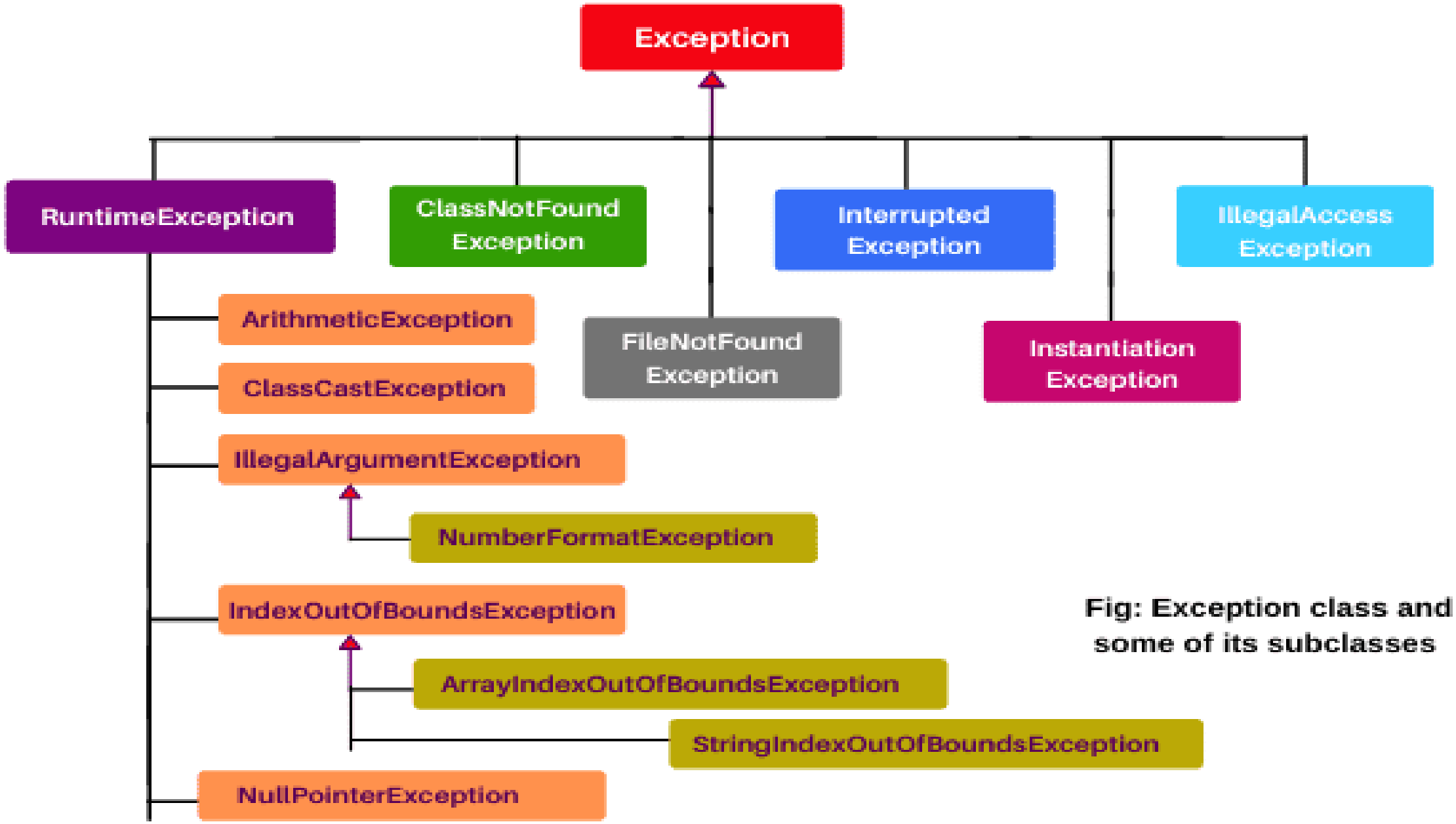


Fig: Exception class and some of its subclasses

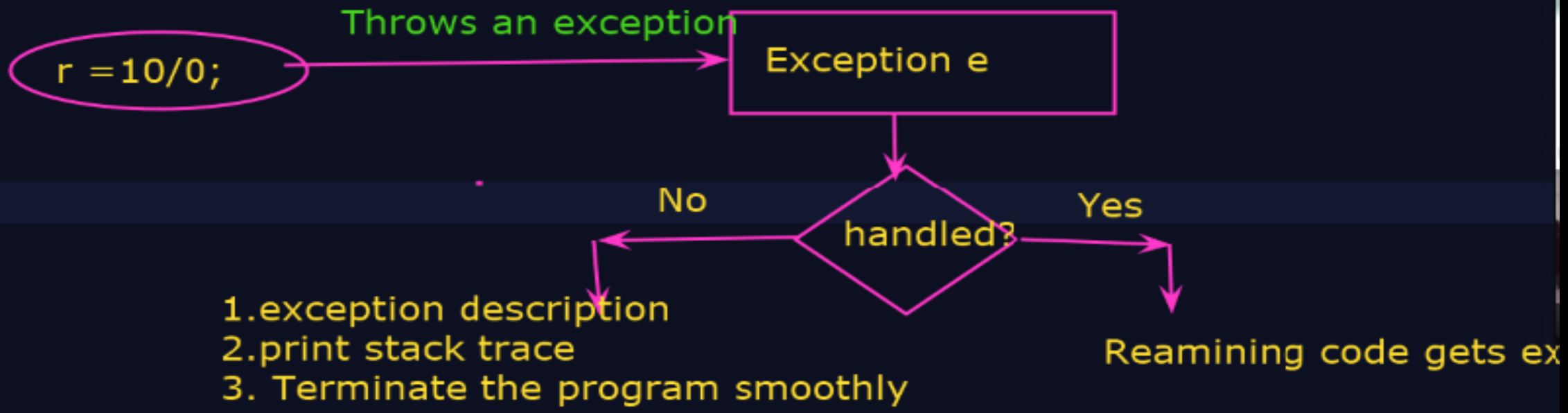
```
try
{
  // A block of code; // generates an exception
}
catch(exception_class var)
{
  // Code to be executed when an exception is
thrown.
}
```

```java
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;
    }
}
```



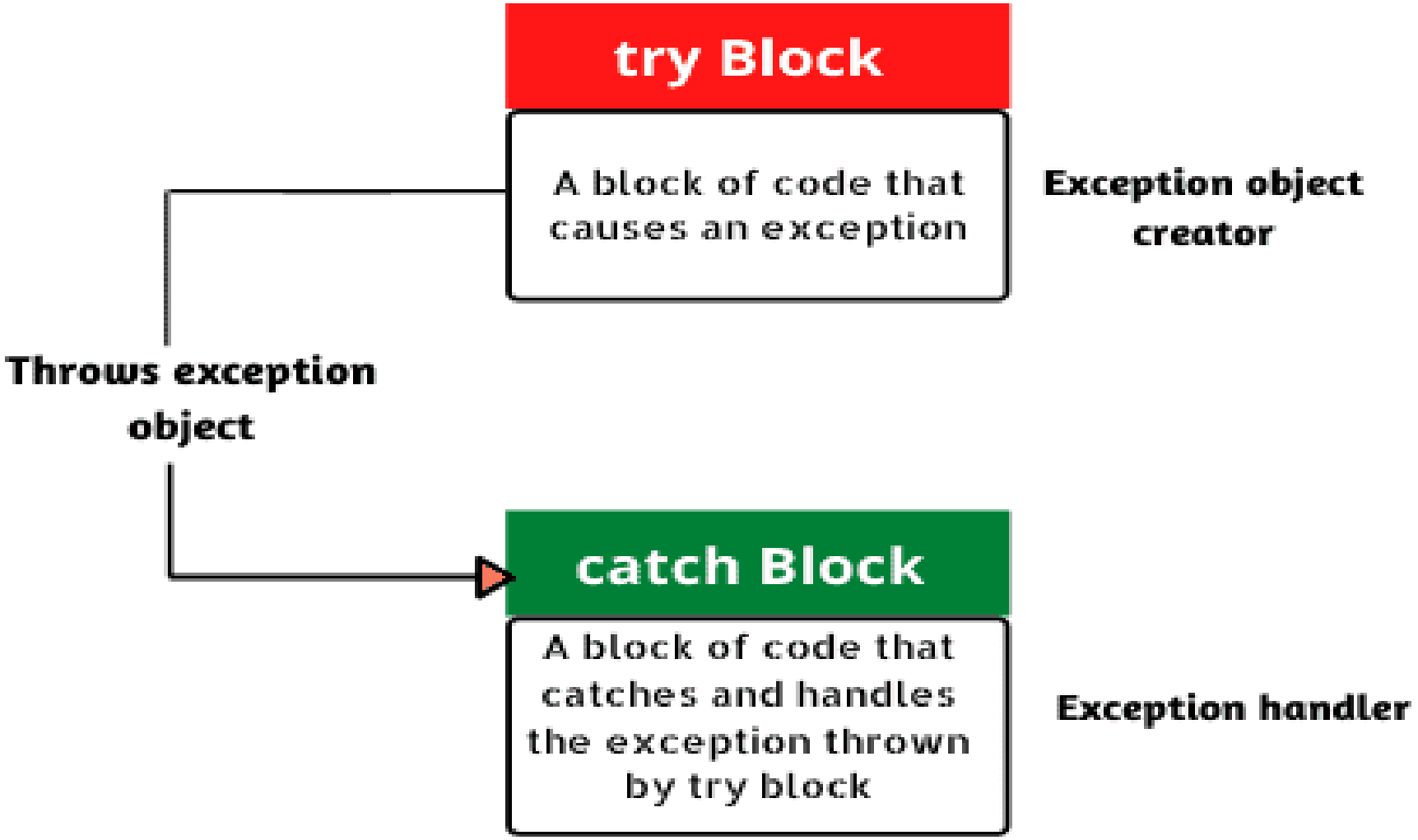


Fig: Exception handling mechanism

```java
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 Exception
Excution started
Cannot divide by zero.
Excution finished

C:\Test>
```

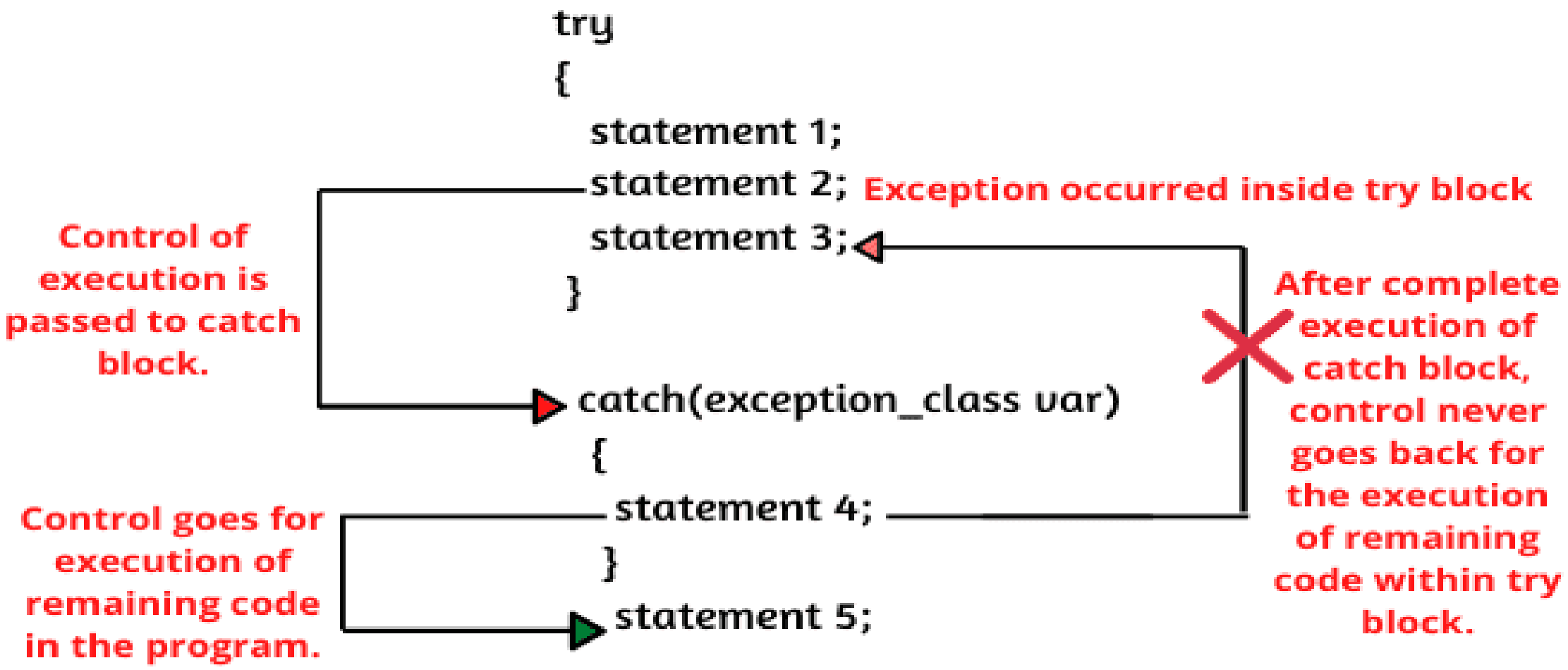


Fig: Control flow of try-catch block in Java

```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 Exceptio

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

C:\Test>
```

```java
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 ExceptionDe

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

C:\Test>
```