OOP Using Java

Faculty: SREEDIVYA

Branch: S6 ECE

History of Java

- Sun Microsystems developed Java in the early 1990s.
- Founder – James Gosling
- First named as ‘Oak’.
- Gosling designed Java with a C/C++ - style syntax that system and application programmers would find familiar.
- Most popular for generating internet – based applications. Hence it is known as the language of internet.

- 1991 - Green Project for consumer electronics market (Oak language → Java)
- 1994 – HotJava Web browser
- 1995 – Sun announces Java
- 1996 – JDK 1.0
- 1997 – JDK 1.1 RMI, AWT, Servlets
- 1998 – Java 1.2 Reflection, Swing, Collections
- 2004 – J2SE 1.5 (Java 5) Generics, enums
- 2014 – Java SE 8 Lambdas - functional programming

- 2017 - Java SE 9
- 2018 - Java SE 10, Java SE 11
- 2019 - Java SE 12, Java SE 13
- 2020 - Java SE 14, Java SE 15
- 2021 - Java SE 16, Java SE 17
- 2022 - Java SE 18, Java SE 19
- 2023 - Java SE 20
- 2024 - Java SE 21
- As of March 2024, Java 22 is the latest version.

Features

- Simple
- Secure
- Portable
- Object-oriented
- Robust
- Multithreaded
- Architecture Neutral
- Interpreted
- Distributed

- **Simple**

There is no complicated statements, no Struct and Union, no pointer usage, no operator overloading, no multiple inheritance.

- **Secure**

Enables construction of virus - free systems and always run in Java Runtime Environment with almost null interaction with system OS.

- **Portable**

Java source code is compiled to an intermediate class file called byte-code and it can be carried to any platform.

- **Object oriented**

Java programming language is composed of objects and classes.

- **Robust**

In a well – written Java program, all run – time errors are handled by the system. Strong type-checking and exception handling mechanism is there.

- **Multithreaded**

Multiple tasks are performed in a single java program by defining multiple threads.

- **Architecture – Neutral**

Independent of any processor type and machine architecture. Java programs are written based on the principle of “Write Once, Run Anywhere”(WORA).

- **Interpreted**

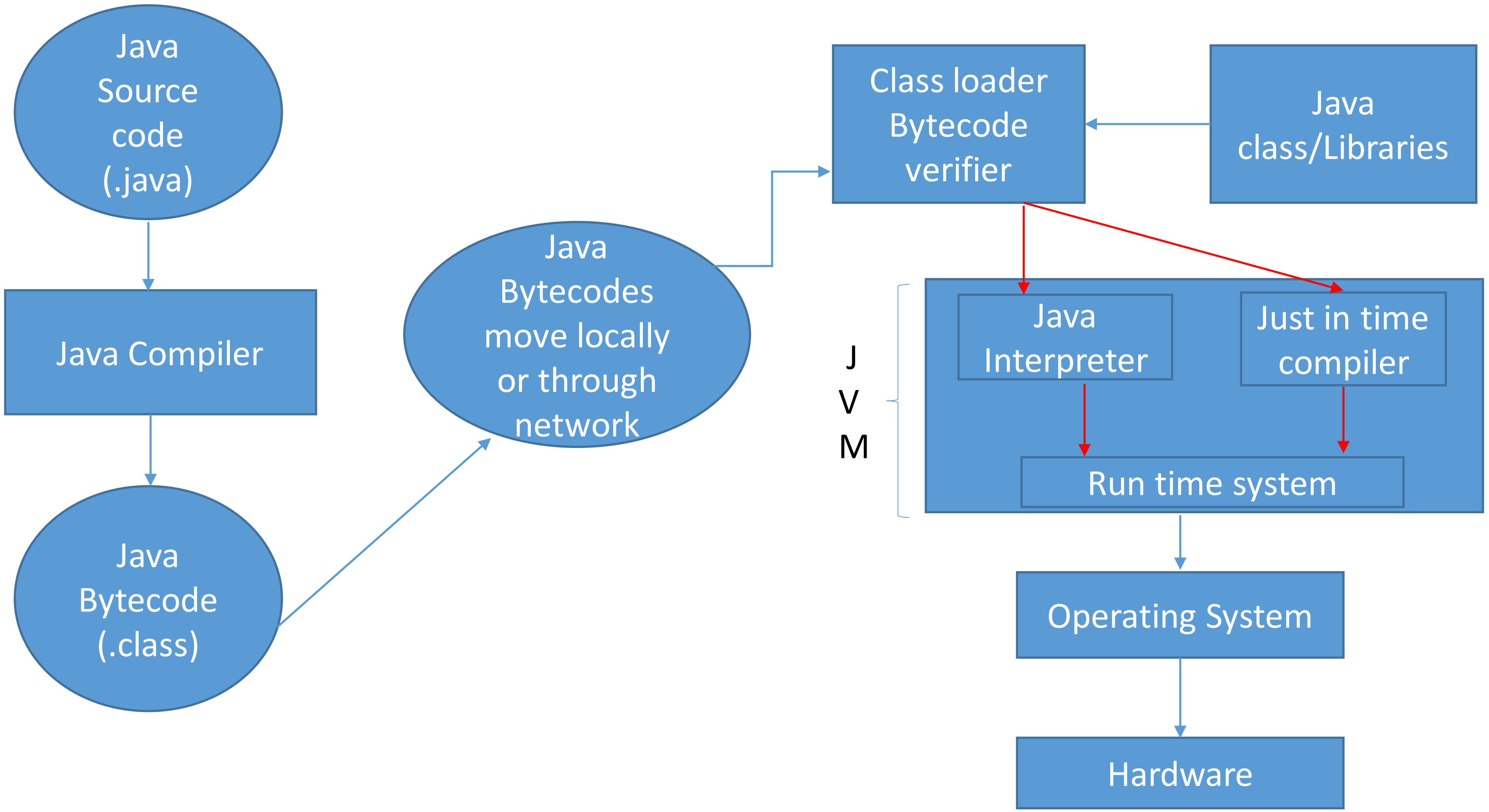
Java source code is first compiled into a byte-code. This byte-code runs on the Java Virtual Machine (JVM), which is usually a software-based interpreter.

- **Distributed**

Designed for the distributed environment of Internet.

Compile-time environment

Run-time environment



Java Source code (.java)

Java Compiler

Java Bytecode (.class)

Java Bytecodes move locally or through network

Class loader Bytecode verifier

Java class/Libraries

J V M

Java Interpreter

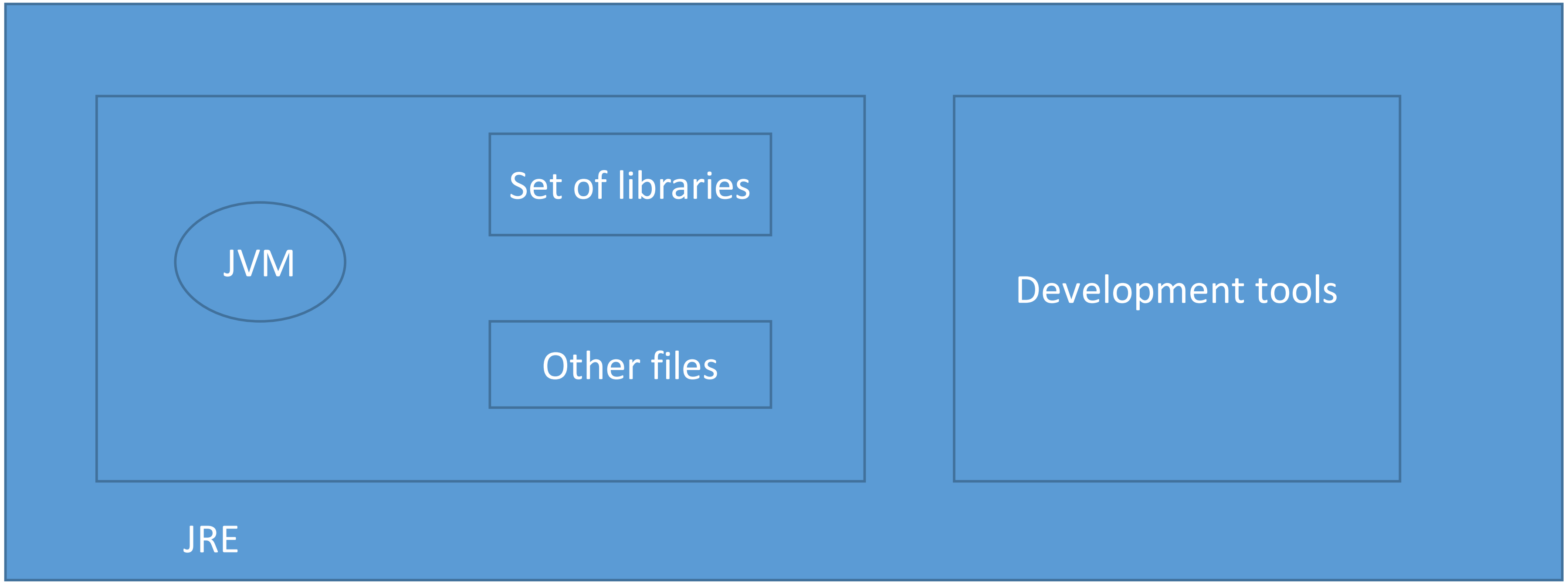
Just in time compiler

Run time system

Operating System

Hardware

Java Platform



JDK

Java Development Kit (JDK)

- JDK contains tools needed to develop the Java programs and JRE to run the programs.

- The tools include Compiler(javac), Java application launcher(java), AppletViewer, The Java Debugger(jdb), Java Documentation Generator(javadoc) etc.

Java Runtime Environment(JRE)

- JRE is made up of a JVM, Java class libraries, and the Java class loader.

- Java Virtual Machine(JVM) executes bytecode. It is available on many types of software and hardware platforms which enables java to function as a platform on its own. JVM also performs garbage collection.

Basic Data types

- A data type tells the compiler:

  - how much memory to allocate

  - format in which to store data

  - type of operations to perform on data

- 8 Primitive data types supported by Java are,

  byte, char, boolean, short, int, long, float, double

- Supports literals(constant values) and strings

- Java uses variables for storing values.
- Supports expressions.
- Supports all arithmetic, increment and decrement, assignment, bitwise, relational, logical operators.
- Control statements such as if, if else, switch, for, while, do while, for each and jumping statements such as break, continue, return are also used in Java.

Type Casting

Process of converting value of a data type to some other data type e.g., int to byte or float to double. There are 2 types:

(1) Implicit typecasting (Widening conversion)

automatic conversion of lower data type to higher data type.

(2) Explicit typecasting (Narrowing conversion)

explicit conversion of higher data type to lower data type.

Java Program to print “HelloWorld”

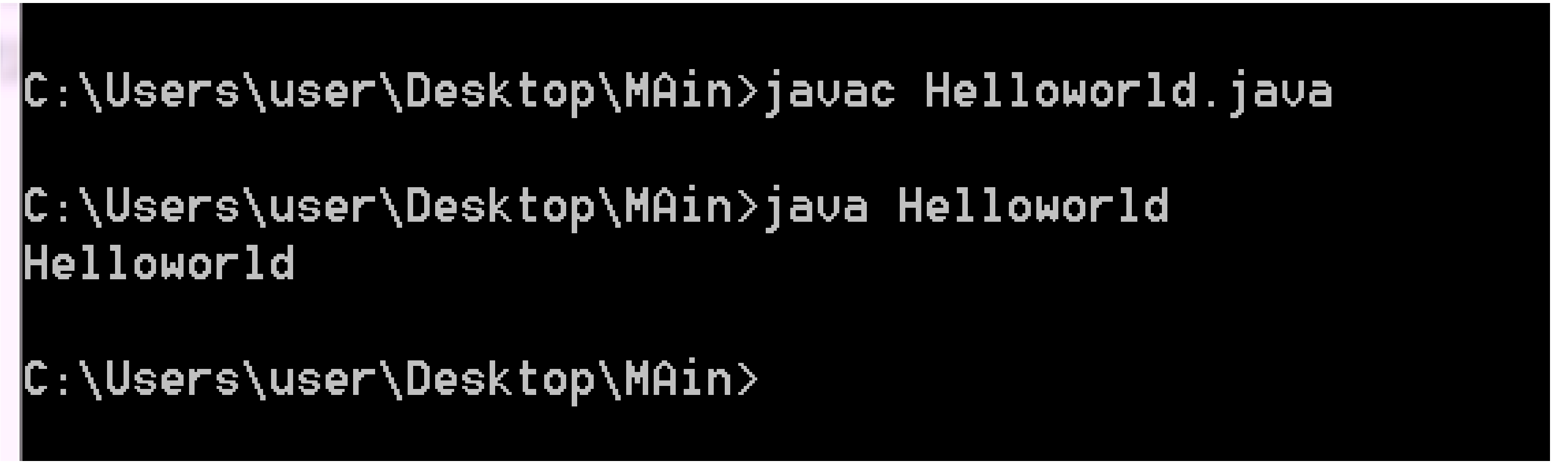
```
public class Helloworld
{
        public static void main(String ar[])
        {
                System.out.println("Helloworld");
        }
}
```

Save the program as Helloworld.java in any folder.

For compilation: javac Helloworld.java

For execution: java Helloworld

OUTPUT



```
C:\Users\user\Desktop\MAin>javac Helloworld.java

C:\Users\user\Desktop\MAin>java Helloworld
Helloworld

C:\Users\user\Desktop\MAin>
```

Classes and objects

- In Java, all code consists of classes.
- Every object is an instance of some class.
- A source program can have multiple classes in it.
- They follow object - oriented concepts such as abstraction, encapsulation, inheritance, polymorphism.
- For inheritance, ‘extends’ keyword is used n Java.
- Objects are created using ‘new’ operator in Java.

```
        class_name   object_name  =  new   constructor_name();
```

Sample program showing object creation in Java

```
public class Main {
  int x = 64;

  public static void main(String ar[]) {
    Main myObj = new Main();
    System.out.println(myObj.x);
  }
}
```

Output

```
64
```

Arrays, Constructors, Strings in Java

- Arrays are collection of similar type.
- It is possible to create arrays with or without objects.
- Constructors are called automatically when an object is created.
- If the class name is same and the number and type of parameters are different, then it is referred to as constructor overloading.
- Strings are considered as objects of type String.
- A string object cannot be modified. In case of modification, a new string object is created.
- String functions present in C, C++ are also present in Java. In addition to it, some new functions are also introduced.

Access control specifiers

- private - can be accessed only within the class.
- default - can be accessed by any other class in the same package.
- protected - can be accessed by any other class in the same package or inherited classes from other package.
- public - can be accessed by any class inside or outside the package.

Static variables, methods, block

Static variables

- can be used to refer the common property of all objects
- gets memory only once at the time of class loading

Static methods

- a method declared static can be called without creating an object
- can directly access other static properties in the same class

Static block

- it is the only block that can be invoked before ‘main()’ method.

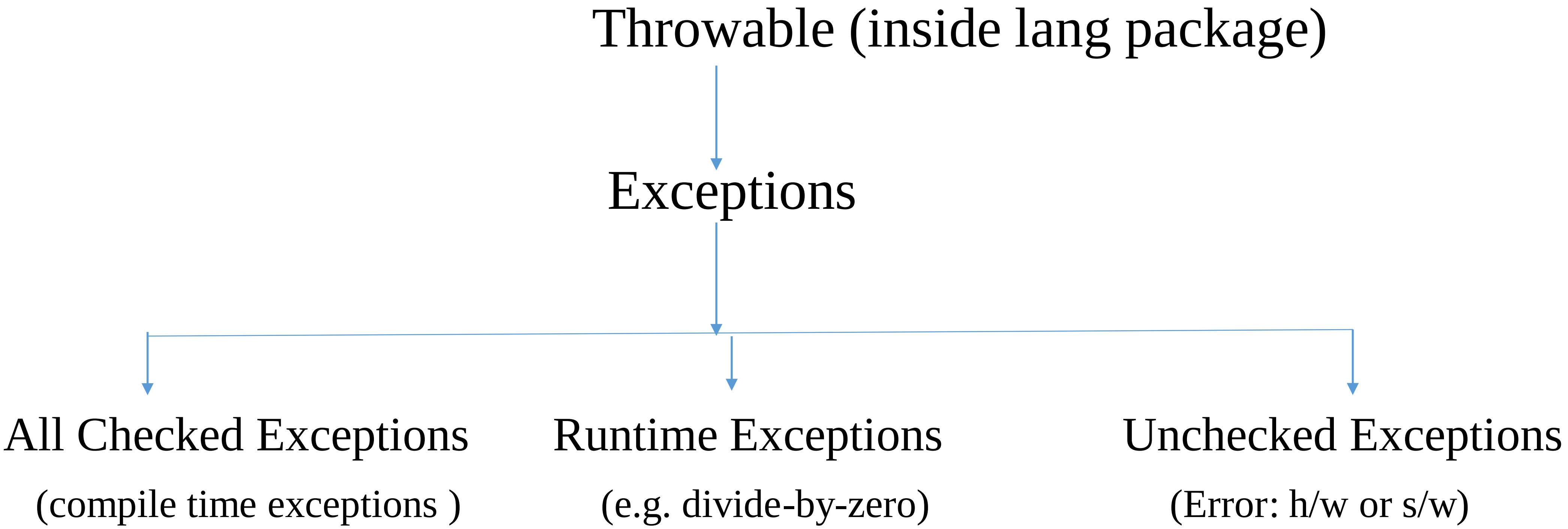
Interfaces

- An abstract class is a class which is not used for creating objects but is a basis for making subclasses.
- Interface is similar to abstract classes, but all methods are abstract and all properties are static final.
- It specifies what a class must do but not how.
- Methods do not have any implementation part.
- Interfaces can be implemented by classes and extended by other interfaces.
- All methods in an interface are public and abstract.
- All variables in an interface are public, static and final.

Exceptions in Java

- Event that occurs during the execution of a program that disrupts the normal flow of instructions is said to be an exception.
- When an error occurs, an exception object is created and handed to the runtime system. This procedure is termed as throwing the exception.
- In Java, the following keywords are used in exception handling

    - try, catch, throw, finally, throws



Throwable (inside lang package)

Exceptions

All Checked Exceptions

(compile time exceptions)

Runtime Exceptions

(e.g. divide-by-zero)

Unchecked Exceptions

(Error: h/w or s/w)

- Programs to be monitored for exceptions are contained within the try block.
- The exception is thrown using the ‘throw’ keyword.
- The corresponding catch block handles the exception.
- The try statement can have multiple catch statements each with a different type of exception.
- ‘finally’- creates a bloc of code that s executed after a try/catch block has completed and before the code following the try/catch block. It is executed whether or not n exception is thrown.
- ‘throws’- lists the types of exceptions that a method might throw.

Multithreading in Java

- The ability of an operating system to execute the different parts of a program called threads at the same time is referred to as multithreading.
- A single process may have multiple threads.
- A thread is a light-weight process.
- All threads share the same memory space, and variables and can communicate with each other directly.
- Primary methods used are:

  start(), run()

Lifecycle of a thread

- New state - create an instance of Thread class.
- Ready state - ready for execution , waiting for CPU access.
- Running state - executing stage of a thread.
- Waiting state - waiting for some action to happen.
- Dead state - when thread has finished execution.

Creating a thread(2 ways)

- Create a thread by extending Thread class
- Create a thread by implementing RunnableInterface

Packages

- It is a container for classes.
- Contains group of related types.
- Mainly used to avoid naming conflicts and to control access.
- If package is not mentioned in a program, then default package will be automatically loaded on to the program.
- Packages in Java begin with java or javax and to avoid conflicts with classes they are all lowercase.
- To have a hierarchy of packages, each package name is separated with a dot(.)

1.java.lang package

- All classes of this package are imported automatically into all programs.
- Contains classes and interfaces that are fundamental to all programs.

Object class

- Super class of all classes.
- All classes inherit the methods of Object class.

Wrapper class

- Used to wrap primitive types in a class structure.
- All primitive types have an equivalent class.
- Includes useful constants and static methods, including one to convert back to the primitive type.

Primitive data type	Wrapper class
double	Double
float	Float
long	Long
int	Integer
short	Short
byte	Byte
char	Character
boolean	Boolean

Number class

- Abstract class
- Superclass of Integer, Long, Float, Double, Short, Byte

System class

- Class for JVM and the control and security for Operating System.
- Define the standard input and output.
- Contains input stream class, output stream class and error stream class.

Math class

- Contains all floating point functions used in geometry and trigonometry and general purpose methods.

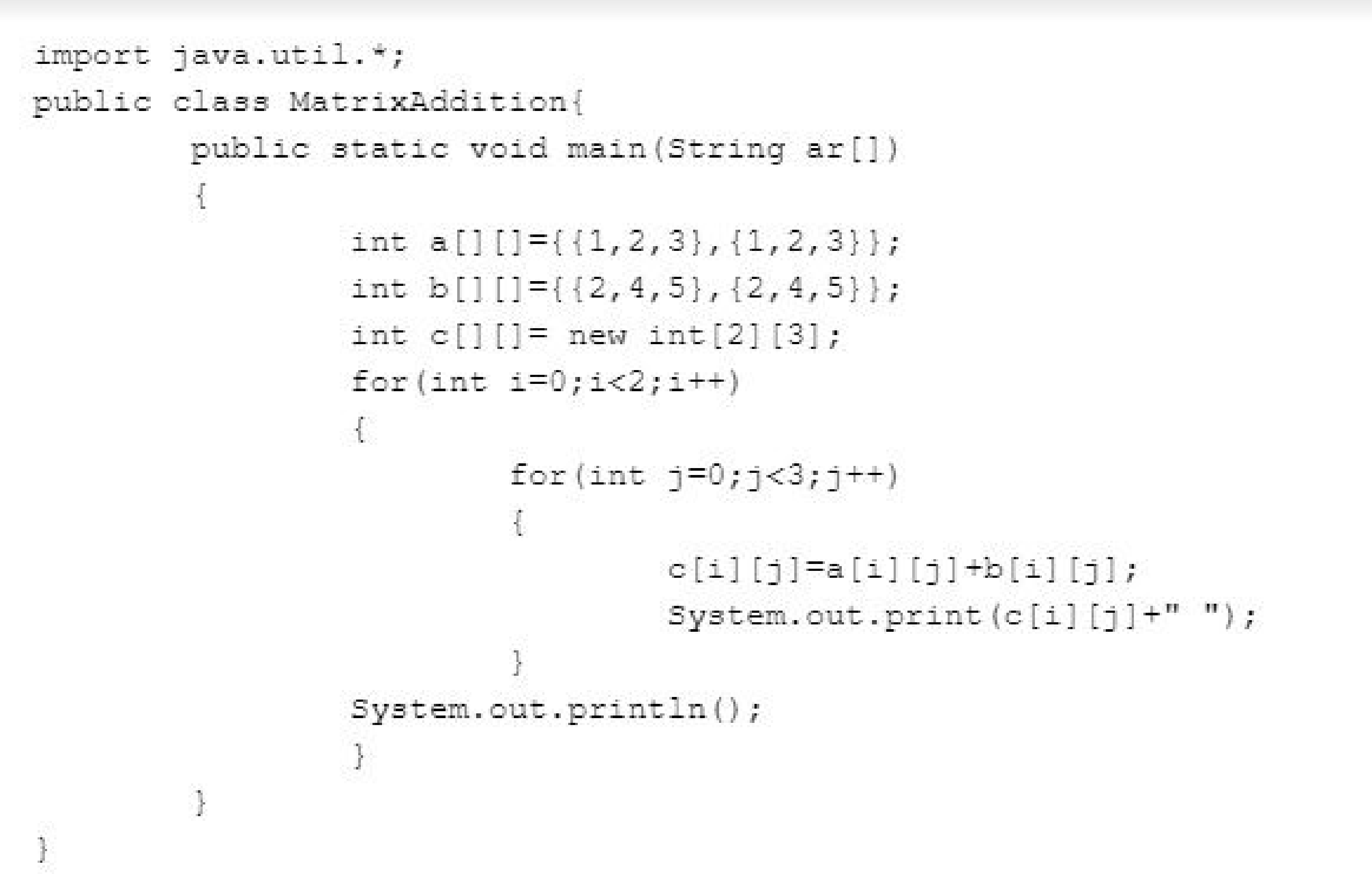
2. java.util package

- A package that supports a wide range of functionalities.
- It includes

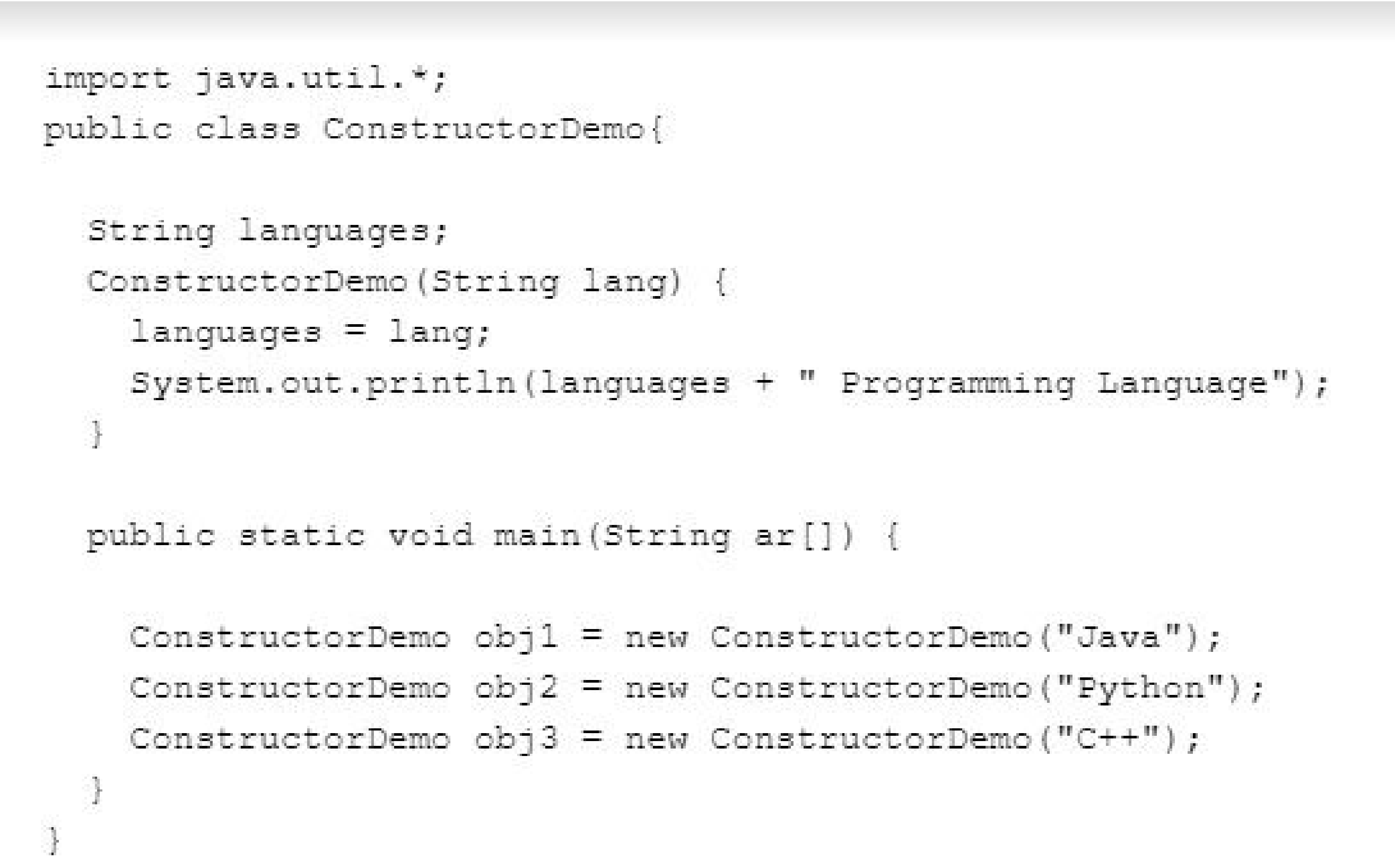
  collections - a group of objects

  collection framework- represents a unified architecture for storing and manipulating group of objects

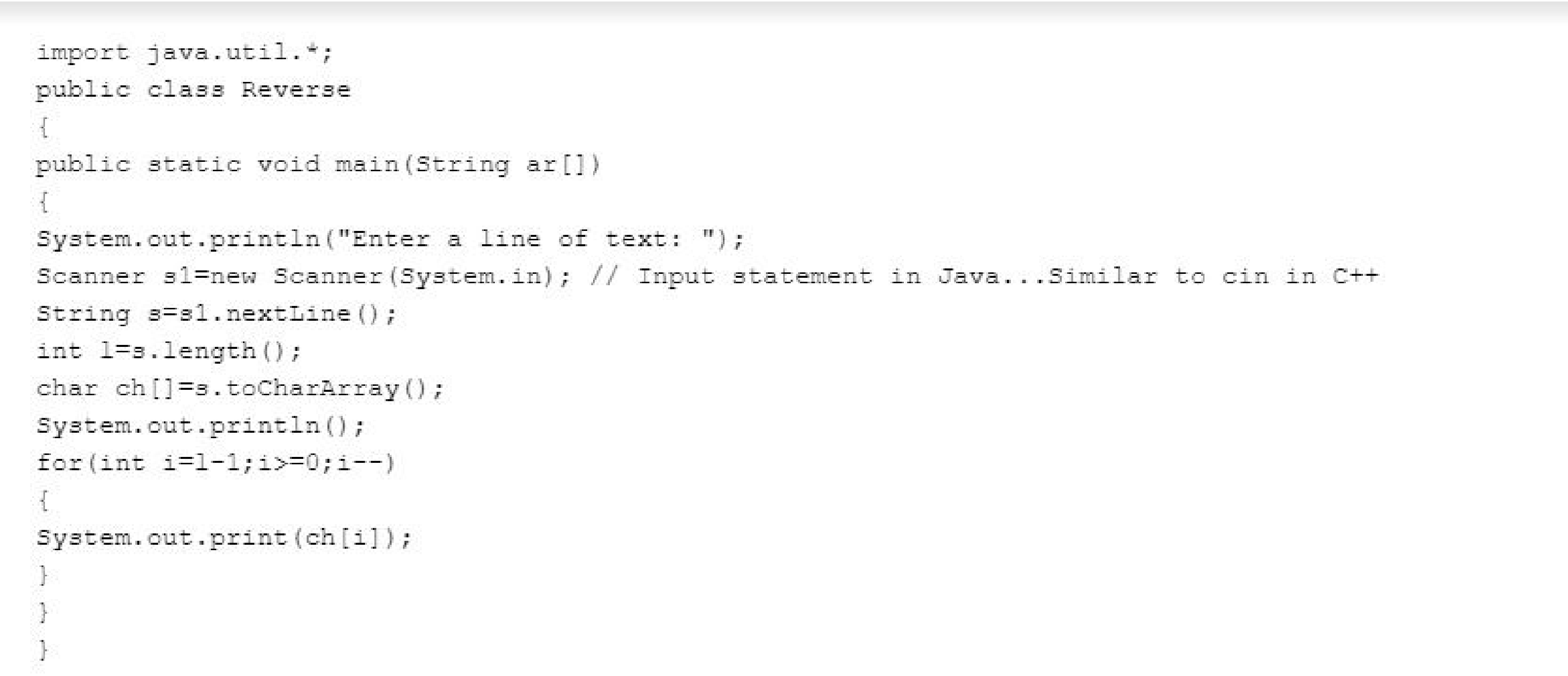
- Collection framework contains interfaces (List Interface, Set Interface, Queue Interface, Map Interface)



```
import java.util.*;
public class MatrixAddition{
        public static void main(String ar[])
        {
                int a[][]={{1,2,3},{1,2,3}};
                int b[][]={{2,4,5},{2,4,5}};
                int c[][]= new int[2][3];
                for(int i=0;i<2;i++)
                {
                        for(int j=0;j<3;j++)
                        {
                                c[i][j]=a[i][j]+b[i][j];
                                System.out.print(c[i][j]+" ");
                        }
                System.out.println();
                }
        }
}
```

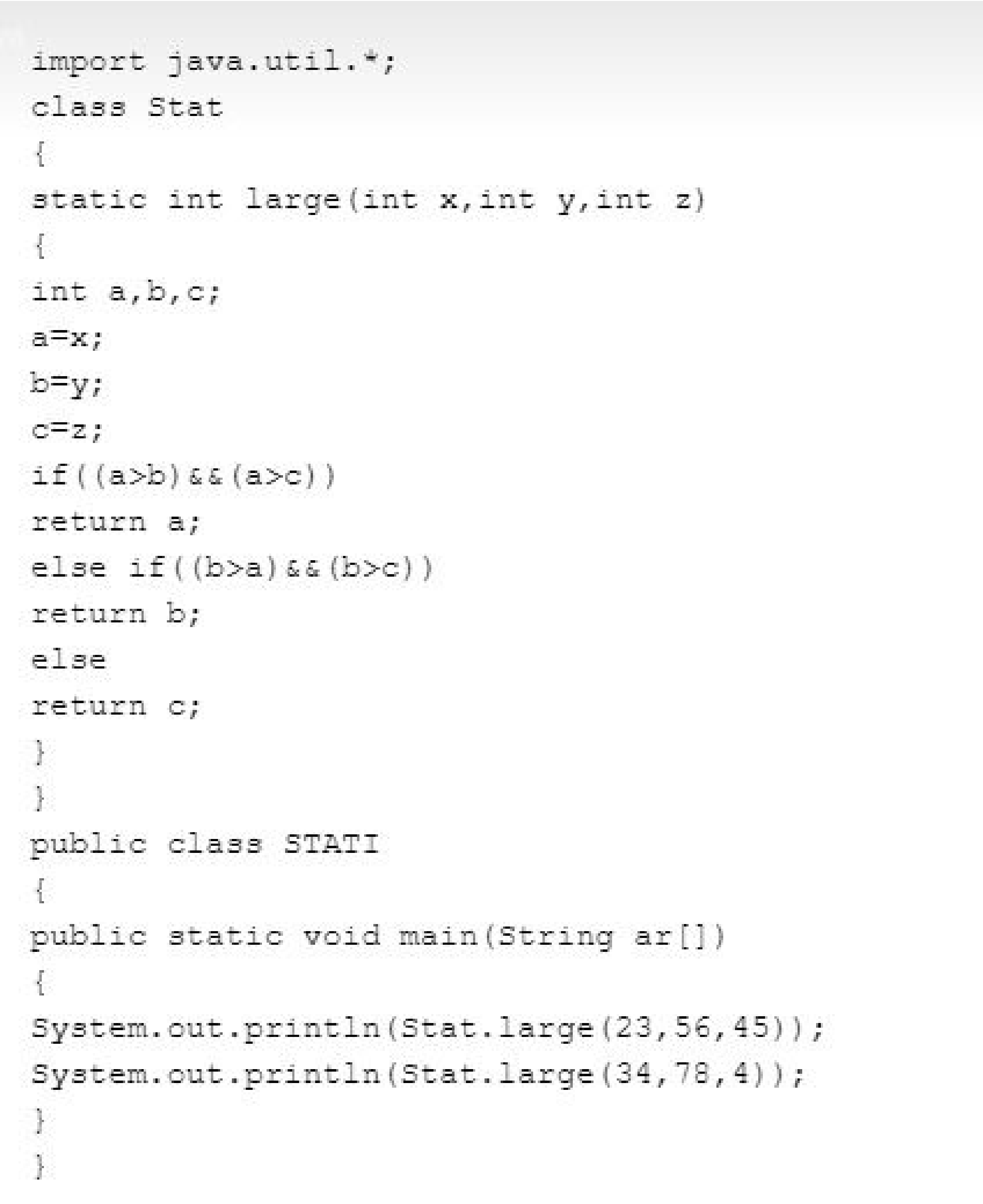


```java
import java.util.*;
public class ConstructorDemo{

  String languages;
  ConstructorDemo(String lang) {
    languages = lang;
    System.out.println(languages + " Programming Language");
  }

  public static void main(String ar[]) {

    ConstructorDemo obj1 = new ConstructorDemo("Java");
    ConstructorDemo obj2 = new ConstructorDemo("Python");
    ConstructorDemo obj3 = new ConstructorDemo("C++");
  }
}
```

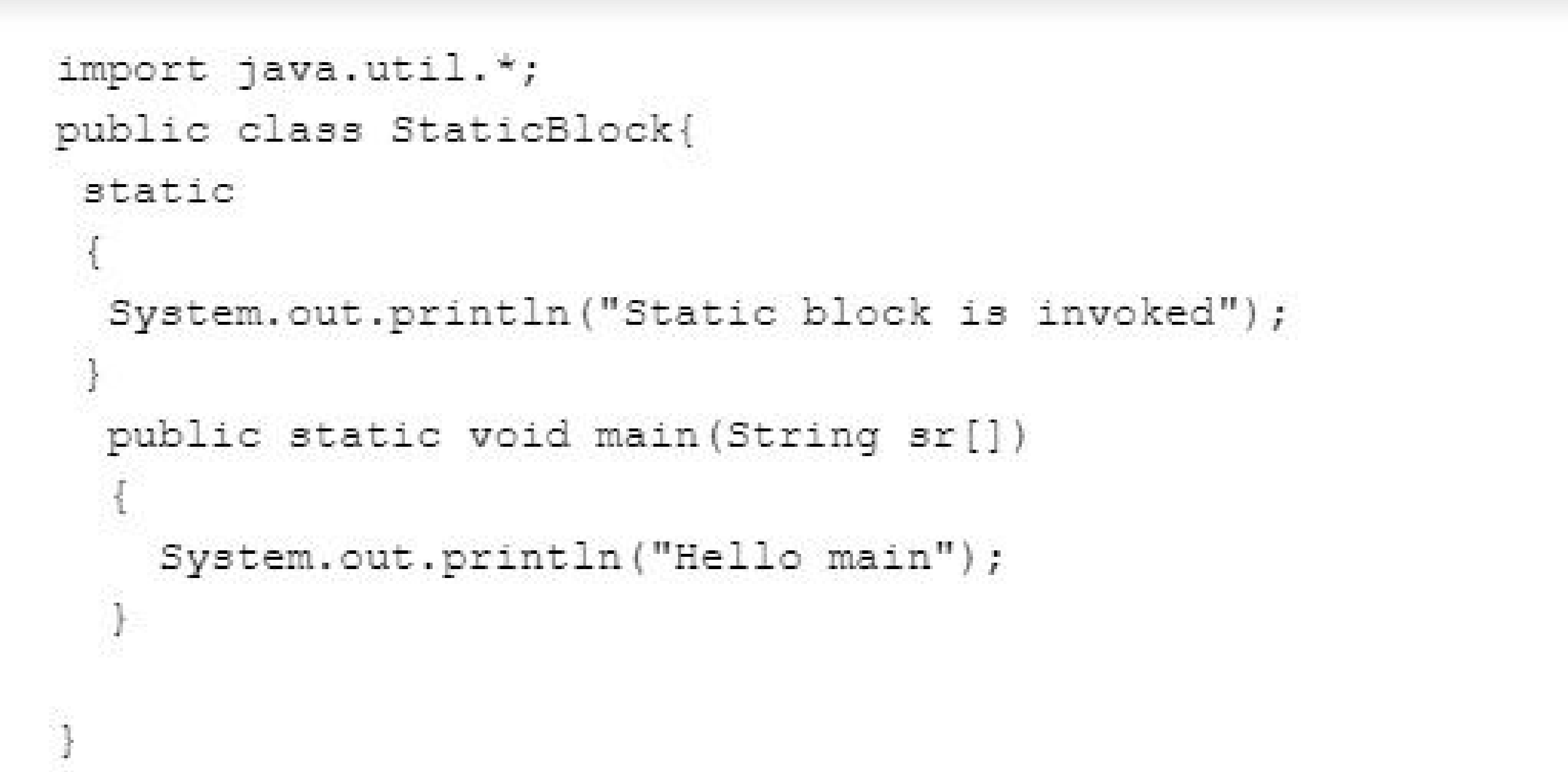


```
import java.util.*;
public class Reverse
{
public static void main(String ar[])
{
System.out.println("Enter a line of text: ");
Scanner s1=new Scanner(System.in); // Input statement in Java...Similar to cin in C++
String s=s1.nextLine();
int l=s.length();
char ch[]=s.toCharArray();
System.out.println();
for(int i=l-1;i>=0;i--)
{
System.out.print(ch[i]);
}
}
}
```

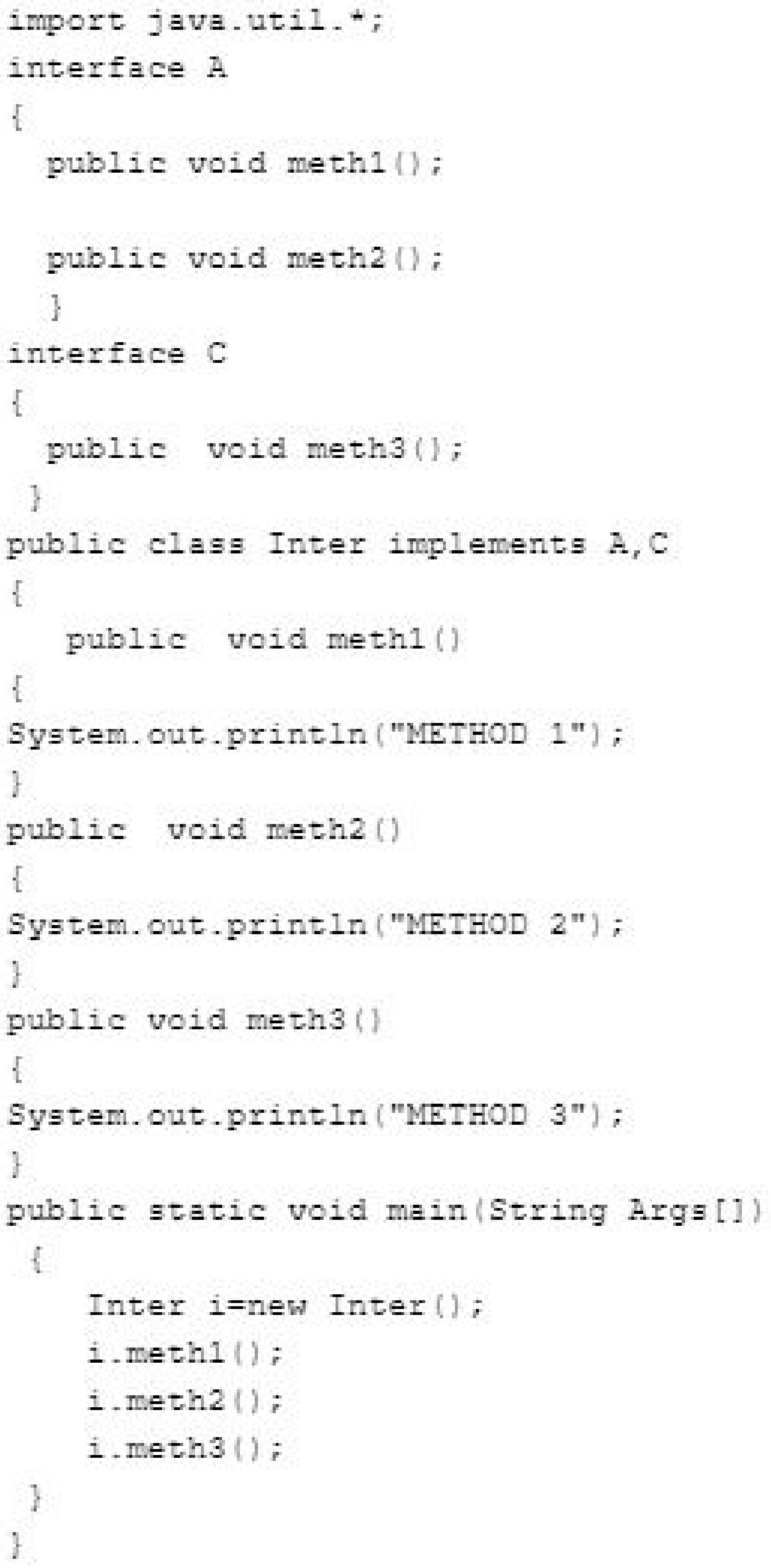
```
import java.util.*;
class StaticVariable{
 int rollno;
 String name;
 static String college = "NIT";
 StaticVariable(int r,String n)
 {
   rollno = r;
   name = n;
 }
 void display()
 {
   System.out.println(rollno+" "+name+" "+college);
 }
}
public class StaticDemo{
  public static void main(String ar[])
  {
   StaticVariable S1 = new StaticVariable(1,"Alice");
   StaticVariable S2 = new StaticVariable(2,"Bob");
   S1.display();
   S2.display();
  }


}
```



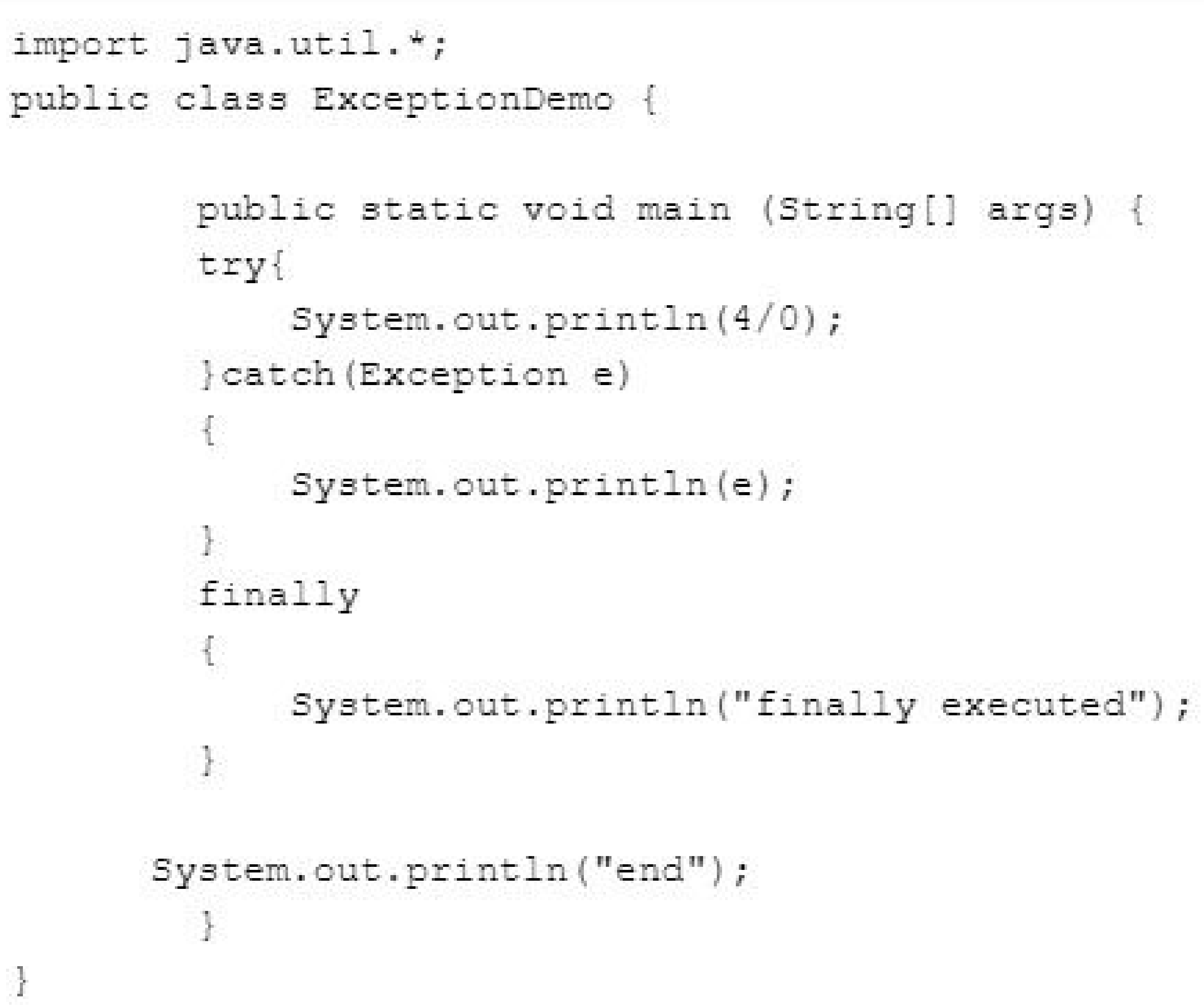
```
import java.util.*;
class Stat
{
static int large(int x,int y,int z)
{
int a,b,c;
a=x;
b=y;
c=z;
if((a>b)&&(a>c))
return a;
else if((b>a)&&(b>c))
return b;
else
return c;
}
}
public class STATI
{
public static void main(String ar[])
{
System.out.println(Stat.large(23,56,45));
System.out.println(Stat.large(34,78,4));
}
}
```



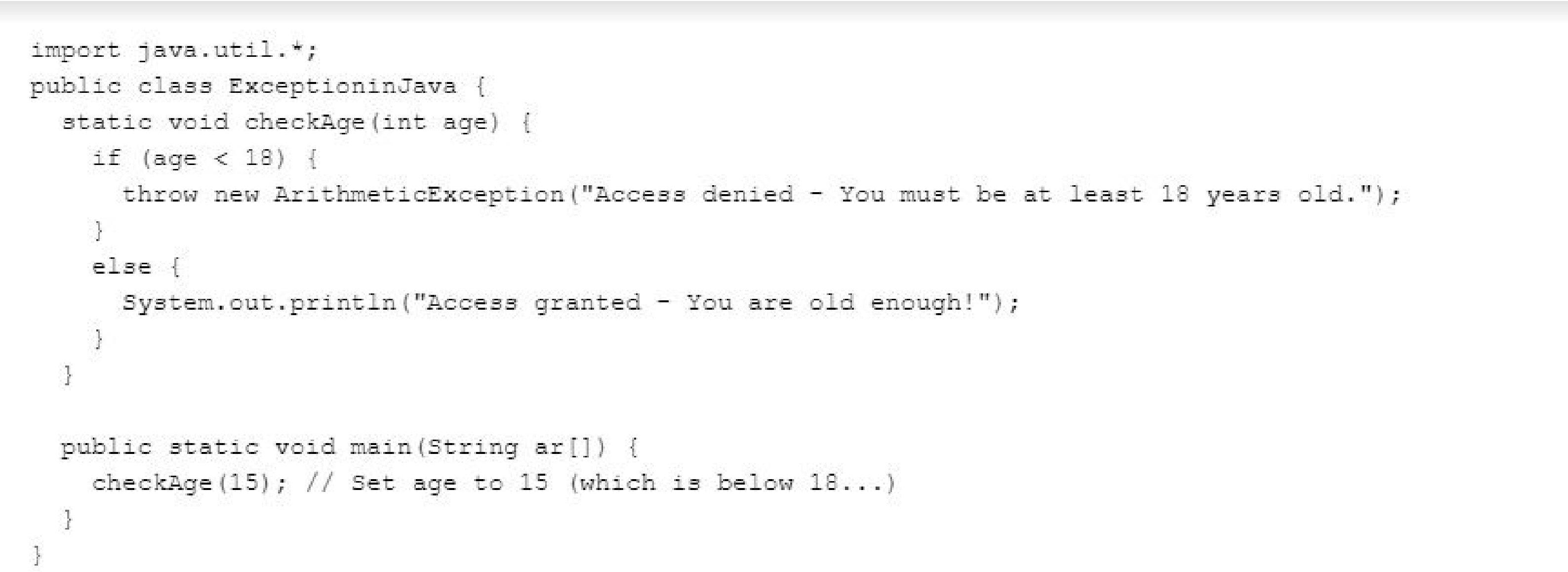
```
import java.util.*;
public class StaticBlock{
 static
 {
  System.out.println("Static block is invoked");
 }
  public static void main(String sr[])
  {
    System.out.println("Hello main");
  }

}
```



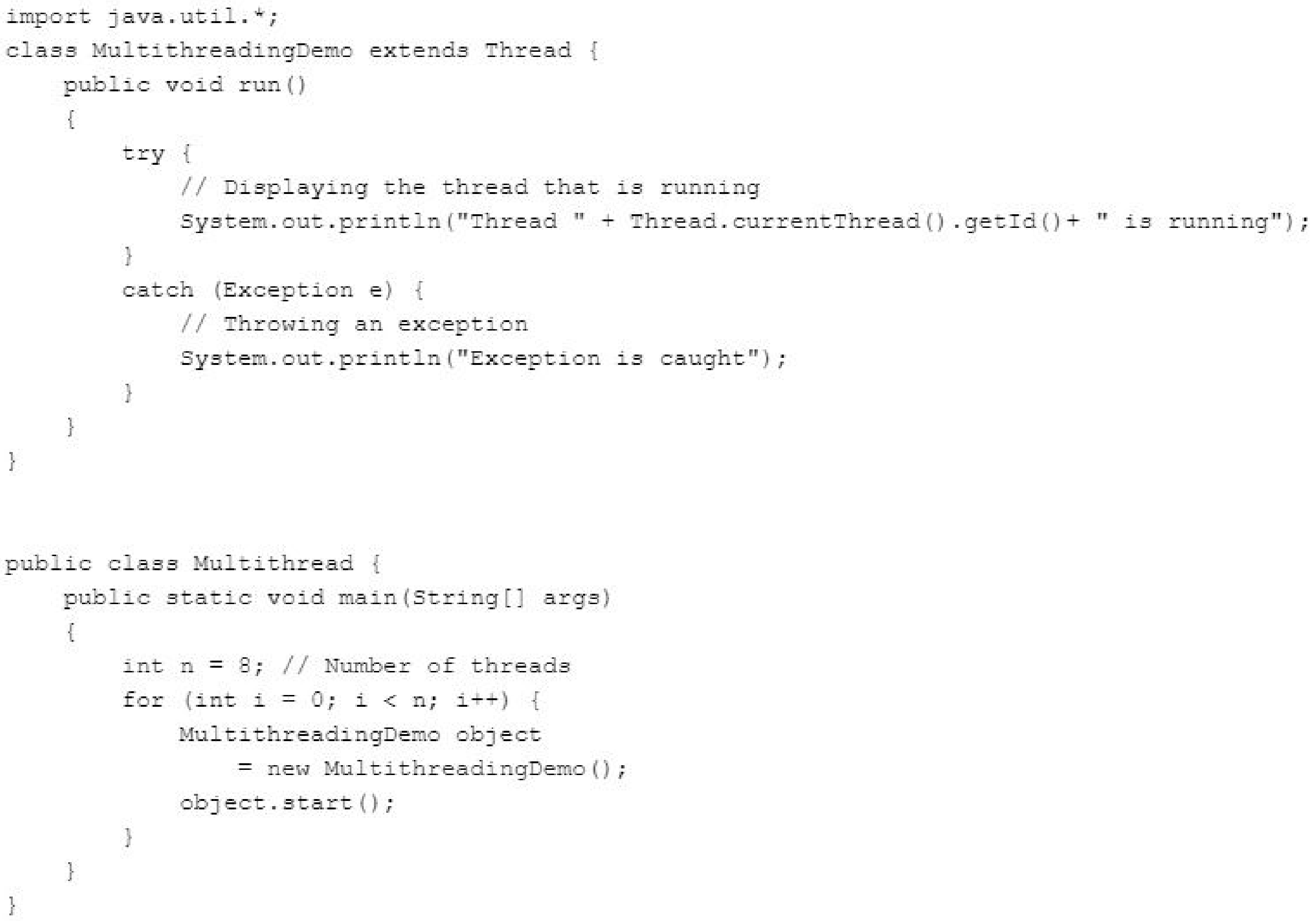
```
import java.util.*;
interface A
{
  public void meth1();

  public void meth2();
  }
interface C
{
  public  void meth3();
 }
public class Inter implements A,C
{
   public  void meth1()
{
System.out.println("METHOD 1");
}
public  void meth2()
{
System.out.println("METHOD 2");
}
public void meth3()
{
System.out.println("METHOD 3");
}
public static void main(String Args[])
 {
    Inter i=new Inter();
    i.meth1();
    i.meth2();
    i.meth3();
 }
}
```



```
import java.util.*;
public class ExceptionDemo {

        public static void main (String[] args) {
        try{
            System.out.println(4/0);
        }catch(Exception e)
        {
            System.out.println(e);
        }
        finally
        {
            System.out.println("finally executed");
        }

      System.out.println("end");
        }
}
```



```
import java.util.*;
public class ExceptioninJava {
  static void checkAge(int age) {
    if (age < 18) {
      throw new ArithmeticException("Access denied - You must be at least 18 years old.");
    }
    else {
      System.out.println("Access granted - You are old enough!");
    }
  }

  public static void main(String ar[]) {
    checkAge(15); // Set age to 15 (which is below 18...)
  }
}
```



```
import java.util.*;
class MultithreadingDemo extends Thread {
    public void run()
    {
        try {
            // Displaying the thread that is running
            System.out.println("Thread " + Thread.currentThread().getId()+ " is running");
        }
        catch (Exception e) {
            // Throwing an exception
            System.out.println("Exception is caught");
        }
    }
}


public class Multithread {
    public static void main(String[] args)
    {
        int n = 8; // Number of threads
        for (int i = 0; i < n; i++) {
            MultithreadingDemo object
                = new MultithreadingDemo();
            object.start();
        }
    }
}
```