Concepts of Programming in Java

- Akshita Chanchlani

Control Statements

Loops

Java has very flexible three looping mechanisms. You can use one of the following three loops:

- while Loop
- do...while Loop
- for Loop

Other Keywords

- Break
- continue

Decision Making

- If
- If..else
- Nested if else
- switch

Wrapper class

- In Java, primitive types are not classes. But for every primitive type, Java has defined a class. It is called wrapper class.

- All wrapper classes are final.

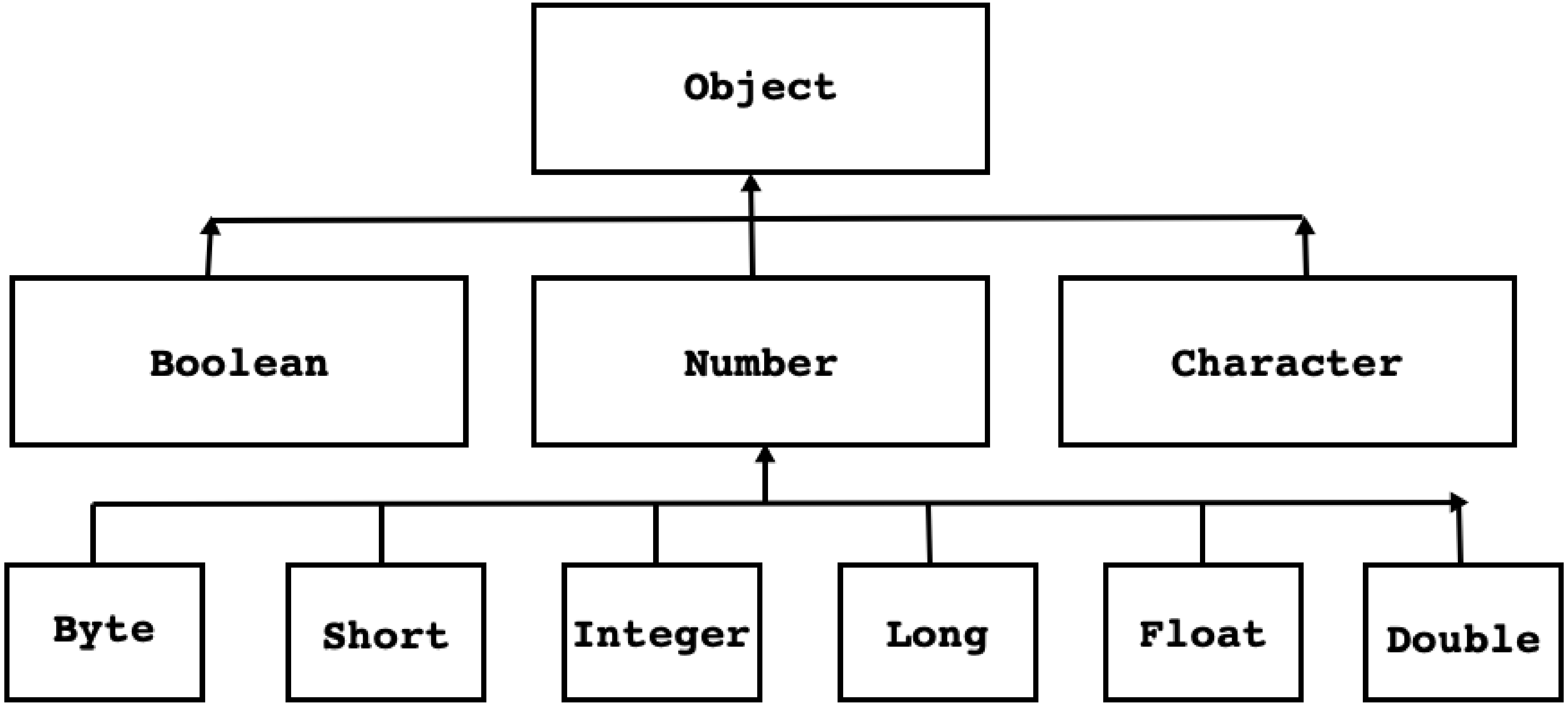
- All wrapper classes are declared in **java.lang** package.

- Uses of Wrapper class
  1. To parse string(i.e. to convert state of string into numeric type).

     example :

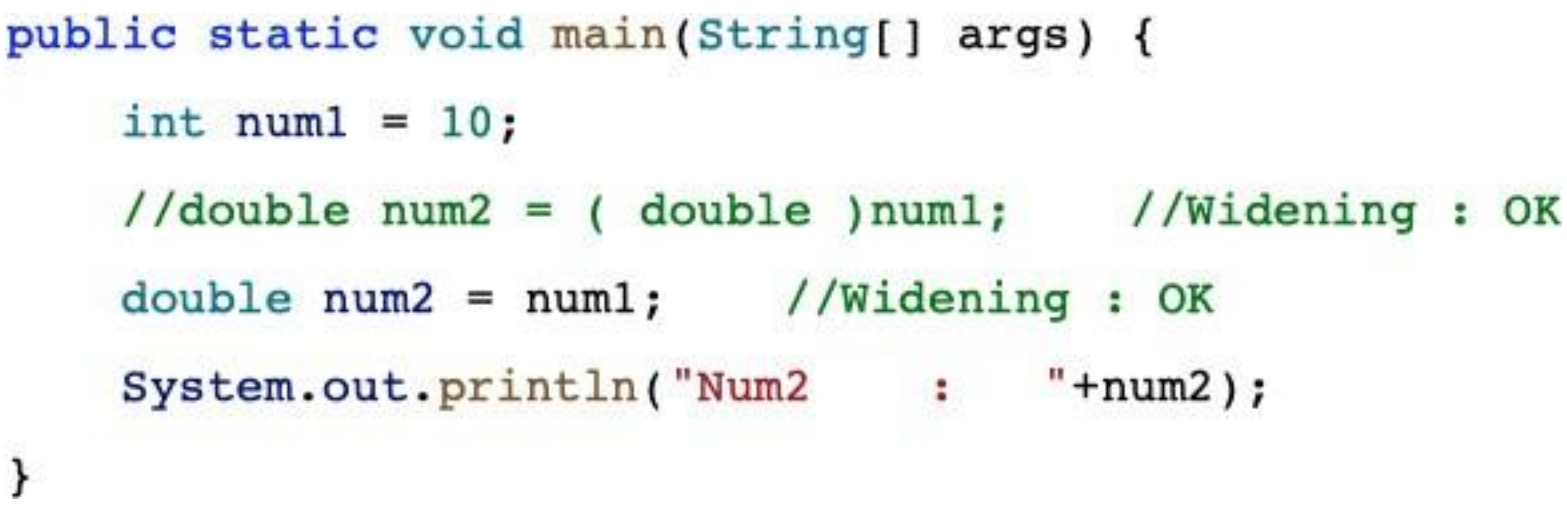
     ```
     int num = Integer.parseInt(“123”)
     float val = Float.parseFloat(“125.34f”);
     double d = Double.parseDouble(“42.3d”);
     ```
  1. To store value of primitive type into instance of generic class, type argument must be wrapper class.
     - ➢ **Stack<int> stk = new Stack<int>( );   //Not OK**
     - ➢ **Stack<Integer> stk = new Stack<Integer>( );   //OK**

Wrapper class



Widening

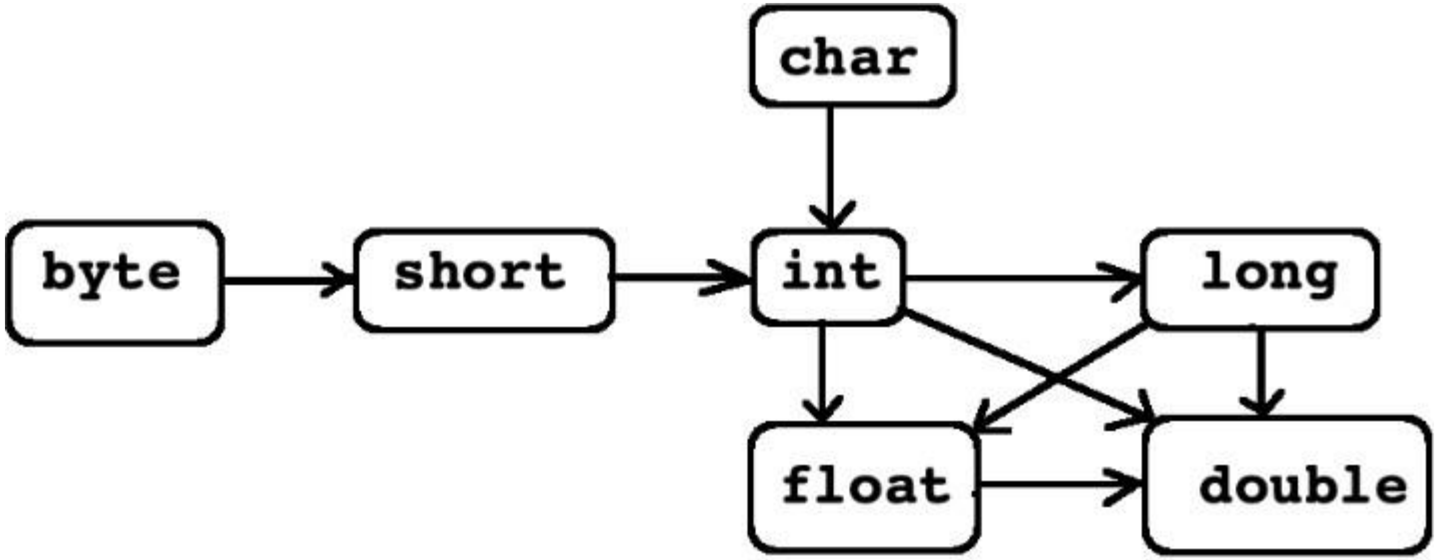
- Process of converting value of variable of narrower type into wider type is called widening.
- E.g. Converting int to double
-



```
public static void main(String[] args) {
    int num1 = 10;
    //double num2 = ( double )num1;    //Widening : OK
    double num2 = num1;    //Widening : OK
    System.out.println("Num2    :    "+num2);
}
```

- In case of widening, there is no loss of data
- So , explicit type casting is optional.

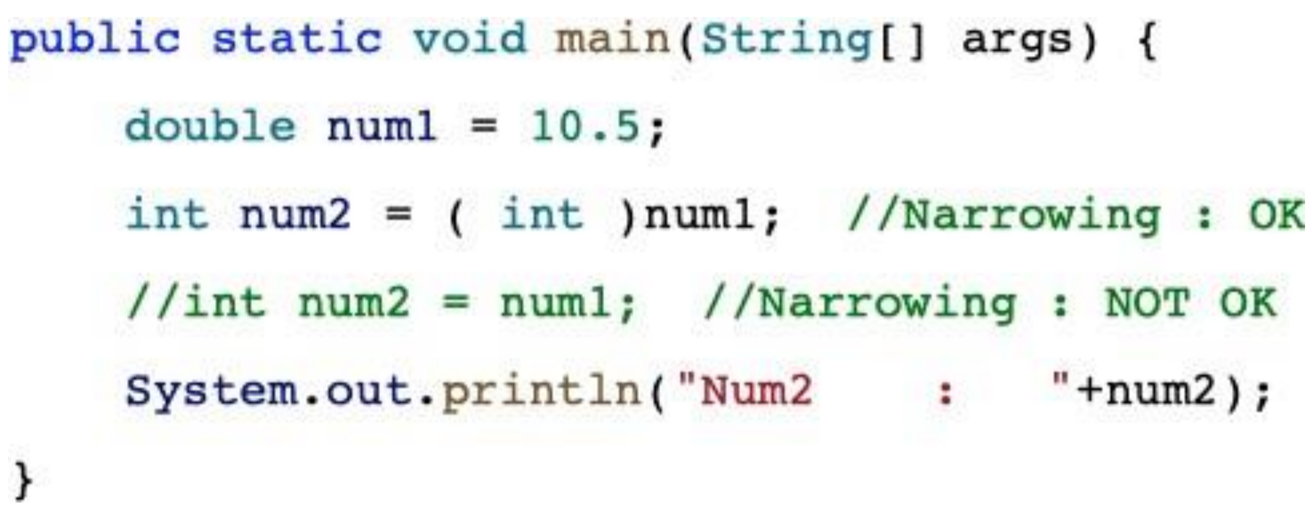
Widening



Widening Conversion

Narrowing

- Process of converting value of variable of wider type into narrower type is called narrowing.

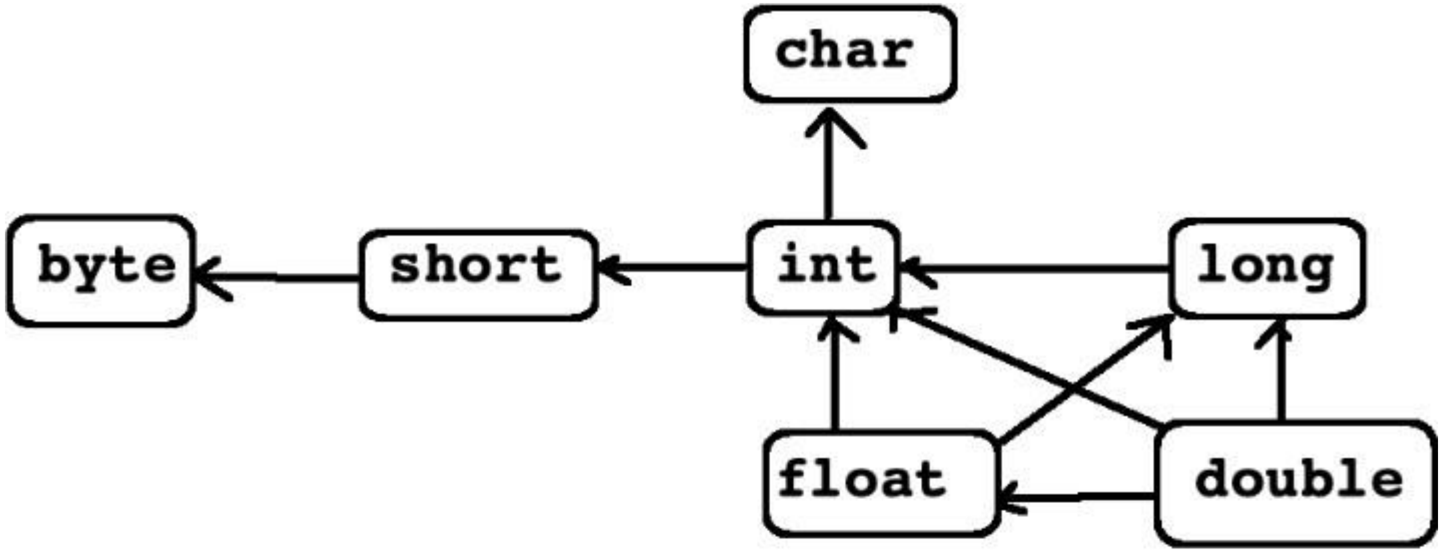


```
public static void main(String[] args) {
    double num1 = 10.5;
    int num2 = ( int )num1;  //Narrowing : OK
    //int num2 = num1;  //Narrowing : NOT OK
    System.out.println("Num2     :     "+num2);
}
```

- In case of narrowing, explicit type casting is mandatory.

Note : In case of narrowing and widening both variables are of primitive

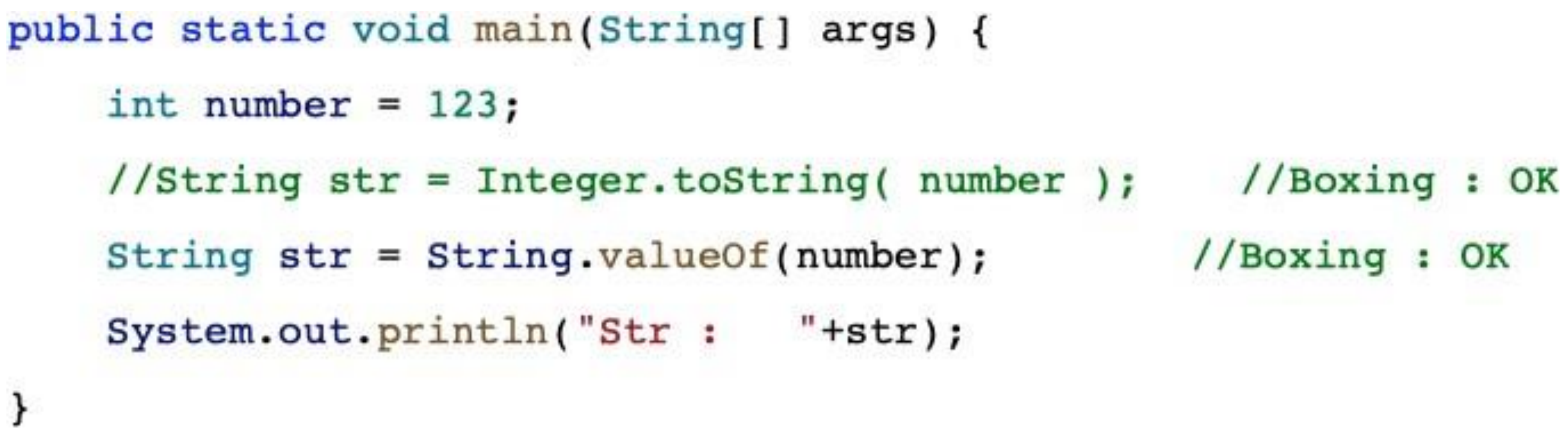
Narrowing



Narrowing Conversion.

Boxing

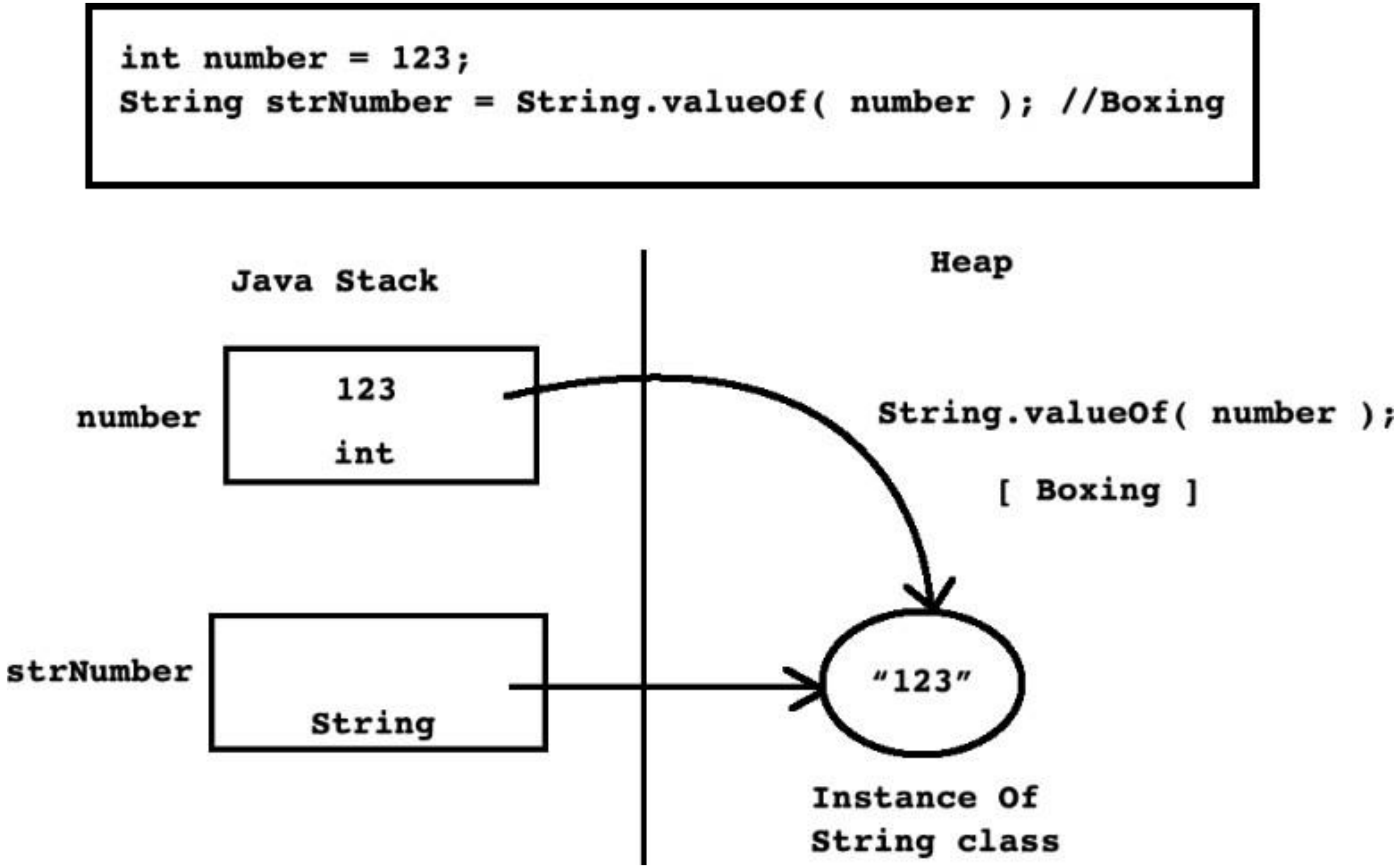
- Process of converting value of variable of primitive type into non primitive type is called **boxing.**



```
public static void main(String[] args) {
    int number = 123;
    //String str = Integer.toString( number );     //Boxing : OK
    String str = String.valueOf(number);        //Boxing : OK
    System.out.println("Str :   "+str);
}
```

- int n1=10; float f=3.5f; double d1=3.45
- String str1=String.valueOf(n1);
- String str2=String.valueOf(f);
- String str3=String.valueOf(d1);

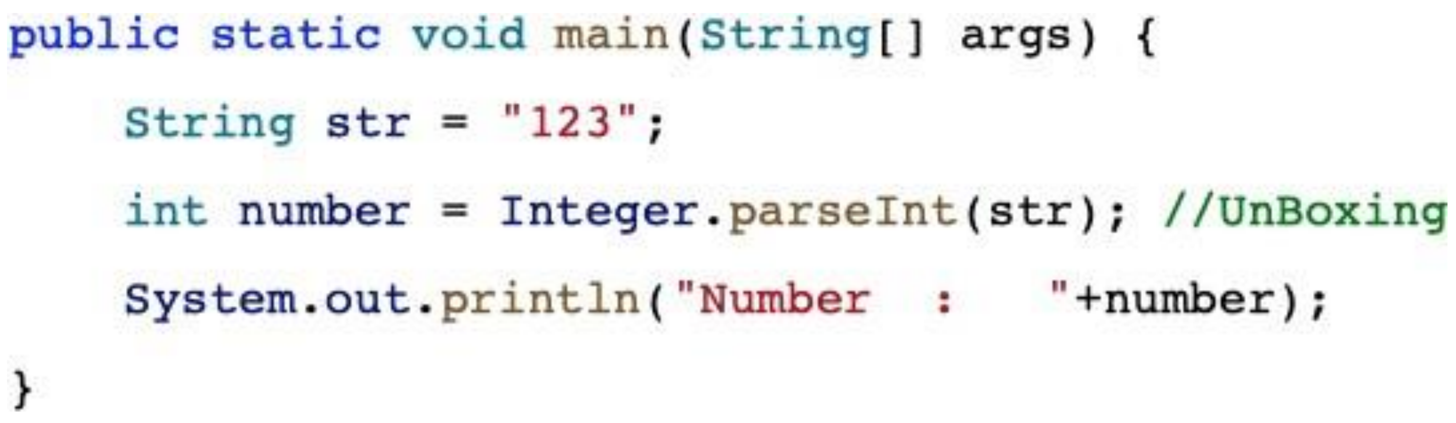
Boxing



```
int number = 123;
String strNumber = String.valueOf( number ); //Boxing
```

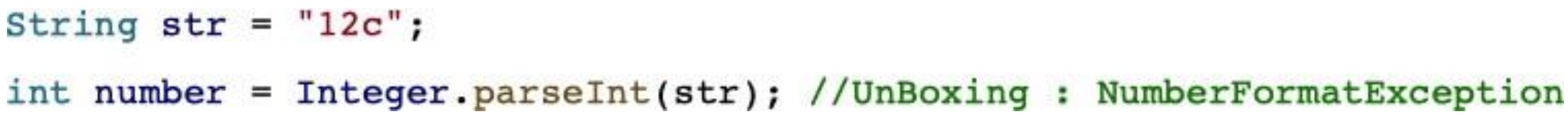
Unboxing

- Process of converting value of variable of non primitive type into primitive type is called unboxing.



```
public static void main(String[] args) {
    String str = "123";
    int number = Integer.parseInt(str); //UnBoxing
    System.out.println("Number  :   "+number);
}
```

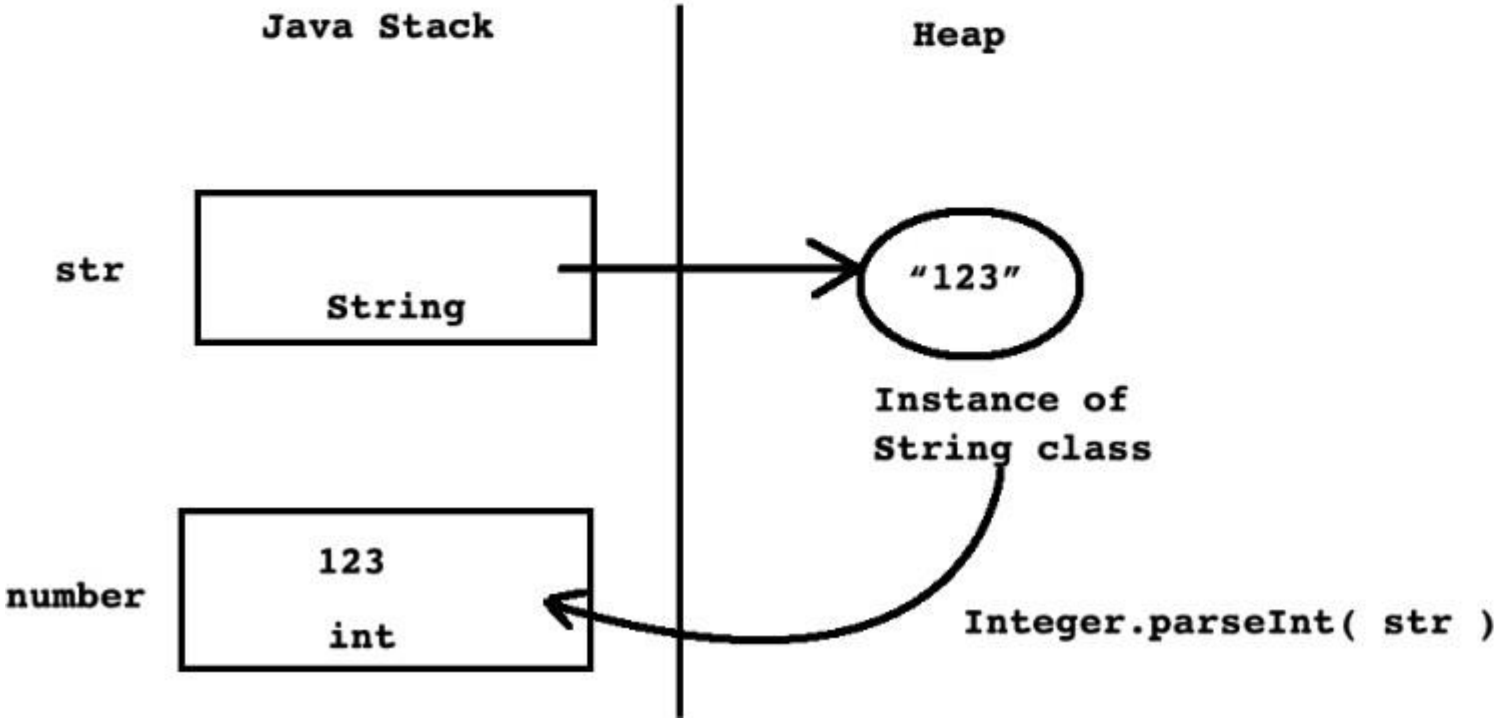
- If string does not contain parseable numeric value then **parseXXX( )** method throws **NumberFormatException.**



```
String str = "12c";
int number = Integer.parseInt(str); //UnBoxing : NumberFormatException
```

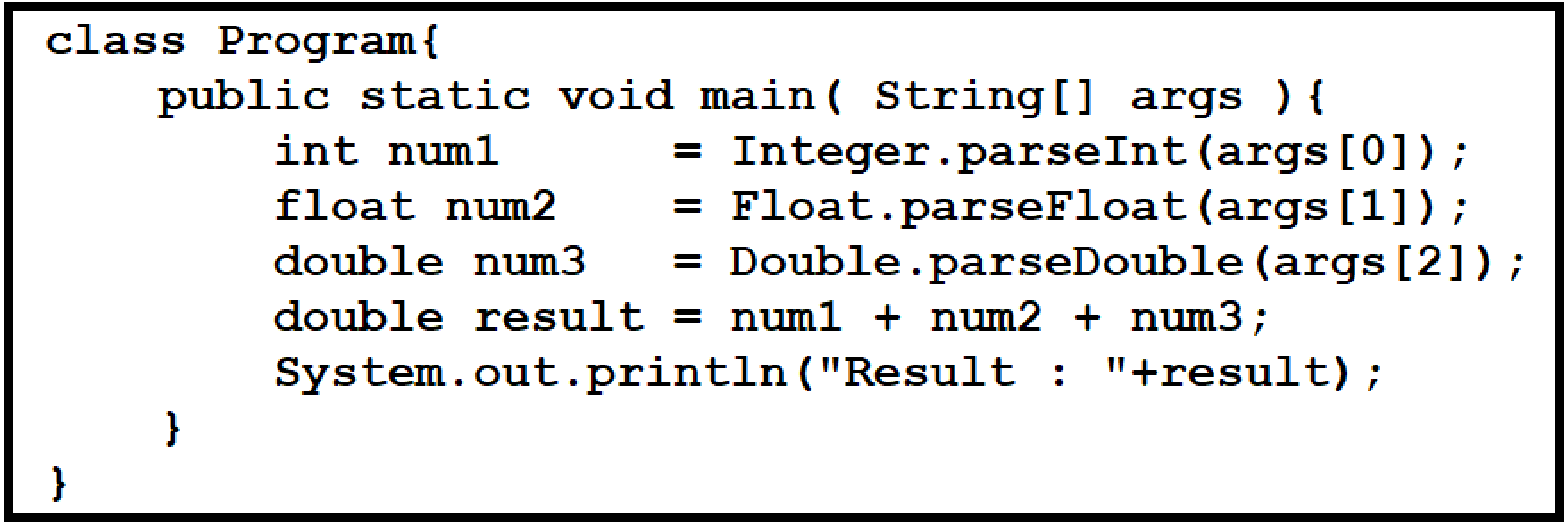
Unboxing

```
String str = “123”;
int number = Integer.parseInt( str ); //UnBoxing
```

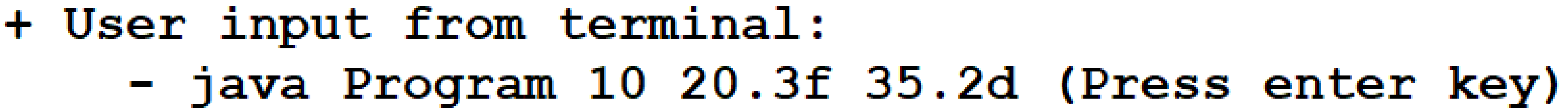


Note : In case of boxing and unboxing one variable is primitive and other Is not primitive

Command line argument



```
class Program{
    public static void main( String[] args ){
        int num1      = Integer.parseInt(args[0]);
        float num2    = Float.parseFloat(args[1]);
        double num3   = Double.parseDouble(args[2]);
        double result = num1 + num2 + num3;
        System.out.println("Result : "+result);
    }
}
```



+ User input from terminal:
    - java Program 10 20.3f 35.2d (Press enter key)

Java Buzzwords

1. Simple
2. Object Oriented
3. Architecture Neutral
4. Portable
5. Robust
6. Multithreaded
7. Dynamic
8. Secure

9.High Performance

10.Distributed

Simple

- Java is **simple** programming language.
  - **Syntax of Java is simpler than syntax of C/C++ hence it is considered as simple**
    - Ø No need of header files and macros.
    - Ø We can not define anything global
    - Ø Do not support structure and union. operator overloading.
    - Ø Do not support copy constructor and assignment operator function constructor member initializer list and default argument constant data member and constant member function.
      Delete operator, destructor , friend function, friend class.
      Multiple class (Multiple inheritance)
    - Ø No diamond problem and virtual base class.
    - Ø Do not support pointer and pointer arithmetic.
  - **Size of software(JDK), that is required to develop Java application is small hence Java is considered as simple programming language.**

Object Oriented

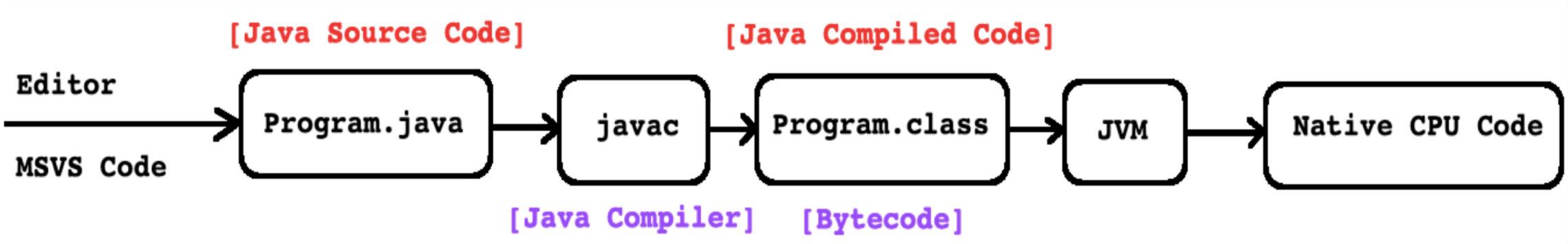
- Java is **object oriented** programming language.
  - Java Supports all the major and minor pillars of oops hence it is considered as object oriented programming language.

  - **Major pillars of oops.**
    1. Abstraction
    2. Encapsulation
    3. Modularity
    4. Hierarchy

  - **Minor pillars of oops.**
    1. Typing / Polymorphism
    2. Concurrency
    3. Persistence.

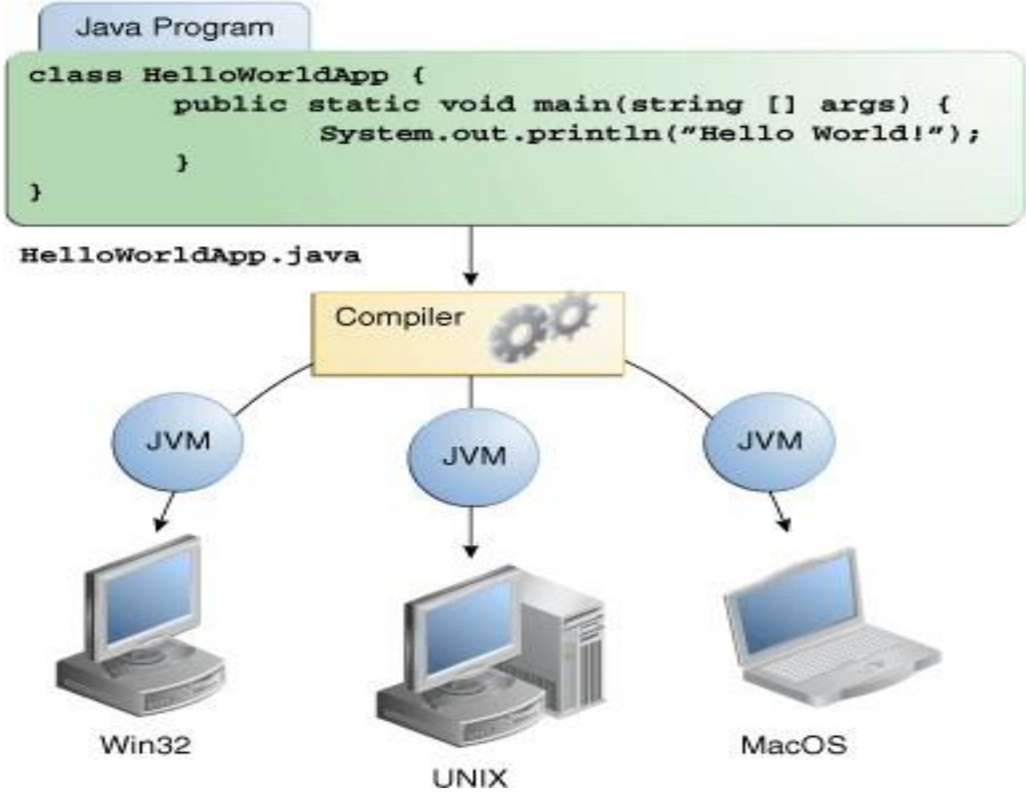
Architecture Neutral

- Java is object **architecture neutral** programming language.
  - Java technology is designed to support applications that will be deployed into heterogeneous network environments. In such environments, applications must be capable of executing on a variety of hardware architectures. Within this variety of hardware platforms, applications must execute on the top of a variety of operating systems. To accommodate the diversity of operating environments, the Java Compilerproduct generates *bytecodes*--an *architecture neutral* intermediate format designed to transport code efficiently to multiple hardware and software platforms.



Portable

- Java is **portable** programming language.
  - o Architecture neutrality is just one part of a truly *portable* system .

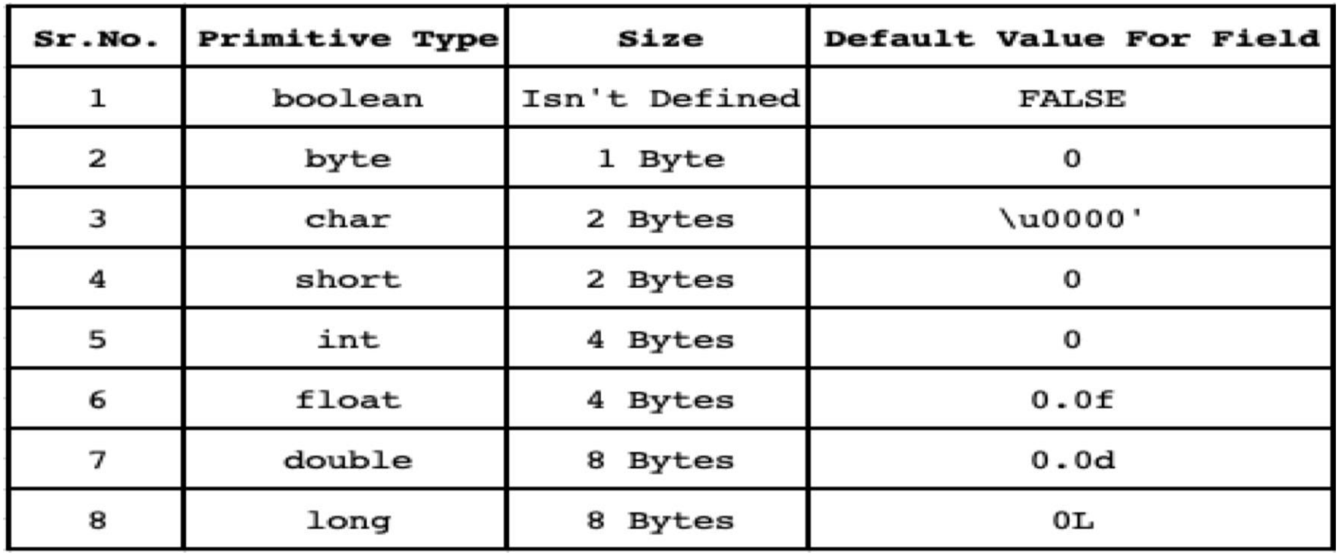


```
class HelloWorldApp {
        public static void main(string [] args) {
                System.out.println("Hello World!");
        }
}
```

Portable

- Java is **portable** programming language.
  - o Java technology takes portability a stage further by being strict in its definition of the basic language.
- o Java data technology puts a stake in the ground and specifies the sizes of its basic types and the behavior of its arithmetic operators.
- o Your programs are the same on every platform--there are no data type

incompatibilities across hardware and software architectures.



Sr.No.	Primitive Type	Size	Default Value For Field
1	boolean	Isn't Defined	FALSE
2	byte	1 Byte	0
3	char	2 Bytes	\u0000'
4	short	2 Bytes	0
5	int	4 Bytes	0
6	float	4 Bytes	0.0f
7	double	8 Bytes	0.0d
8	long	8 Bytes	0L

Robust

- Java is **robust** programming language.
  - o The Java programming language is designed for creating highly *reliable* software. It provides extensive compile-time checking, followed by a second level of run- time checking.
  - o Java is robust because of following features:

1. *Architecture Neutral.*
   - Ø Java developer is free from developing H/W or OS specific coding.
2. *Object orientation.*
   - Ø Reusability reduces developer's effort.
3. *Automatic memory management.*
   - Ø Developer need not to worry about memory leakage / program crashes.
4. *Exception handling.*
   - Ø Java compiler helps developer to provide try-catch block.

Multithreaded

- Java is **multithreaded** programming language.
    - When we start execution of Java application then JVM starts execution threads hence Java is considered as multithreaded.
        1. Main thread
            - Ø It is user thread / non daemon thread.
            - Ø It is responsible for invoking main method.
            - Ø Its default priority is 5(Thread.NORM_PRIORITY).
        2. Garbage Collector / Finalizer
            - Ø It is daemon thread / background thread.
            - Ø It is responsible for releasing / deallocating memory of unused objects.
            - Ø Its default priority is 8(Thread.NORM_PRIORITY + 3).
    - The Java platform supports multithreading at the language level with the addition of sophisticated synchronization primitives: the language library provides the Thread class, and the run-time system provides monitor and condition lock primitives. At the library level, moreover, Java technology's high-level system libraries have been written to be thread safe: the functionality provided by the libraries is available without conflict to multiple concurrent threads of execution.

Dynamic

- Java is **dynamic** programming language.
  - While the Java Compiler is strict in its compile-time static checking, the language and run-time system are *dynamic* in their linking stages. Classes are linked only as needed. New code modules can be linked in on demand from a variety of sources, even from sources across a network.
  - Java is designed to adapt to an evolving environment.
  - Libraries can freely add new methods and instance variables without any effect on their clients.
  - In Java finding out runtime type information is straightforward.
  - In Java, all the methods are by default virtual.

Secure

- Java is **secure** programming language.
  - Java is intended to be used in networked/distributed environments. Toward that end, a lot of emphasis has been placed on security. Java enables the construction of virus-free, tamper-free systems.
  - From the beginning, Java was designed to make certain kinds of attacks impossible, among them:
    1. Overrunning the runtime stack—a common attack of worms and viruses
    2. Corrupting memory outside its own process space
    3. Reading or writing files without permission

High Performance

- Java

  is **high performance** programming language.

  - The Java platform achieves superior performance by adopting a scheme by which the interpreter can run at full speed without needing to check the run-time environment.
  - The *automatic garbage collector* runs as a low-priority background thread, ensuring a high probability that memory is available when required, leading to better performance.
  - Applications requiring large amounts of compute power can be designed such that compute-intensive sections can be rewritten in native machine code as required and interfaced with the Java platform.
  - In general, users perceive that interactive applications respond quickly even though they're interpreted.

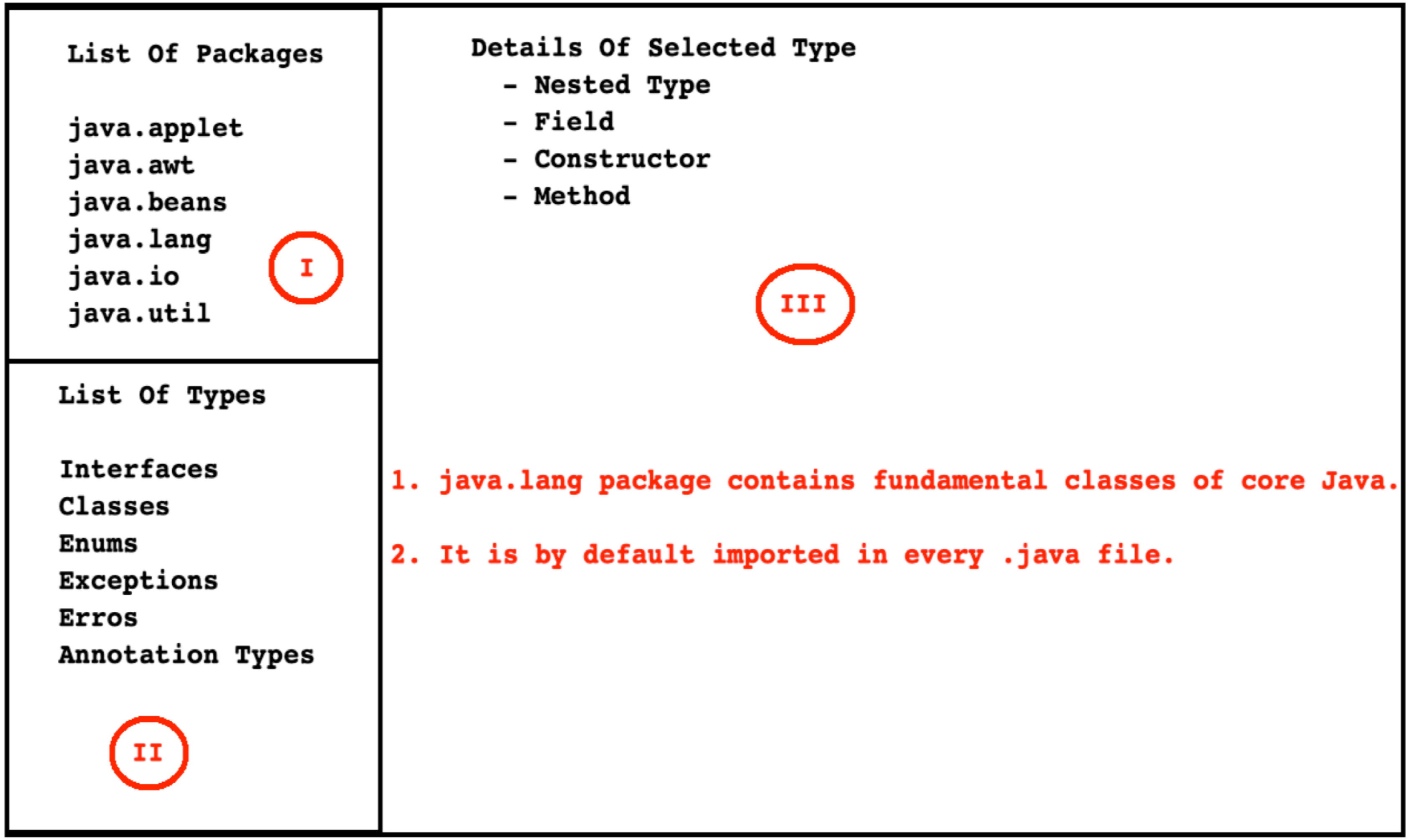
Distributed

- Java is **distributed** programming language.

- Java has an extensive library of routines for coping with protocols like HTTP , TCP/IP and FTP.
- Java applications can open and access objects across the Net via URL with the same ease as when accessing a local file system.

  - Nowadays, one takes this for granted, but in 1995, connecting to a web server from a C++ or Visual Basic program was a major undertaking.

Java DOCS



List Of Packages

java.applet
java.awt
java.beans
java.lang
java.io
java.util

I

List Of Types

Interfaces
Classes
Enums
Exceptions
Erros
Annotation Types

II

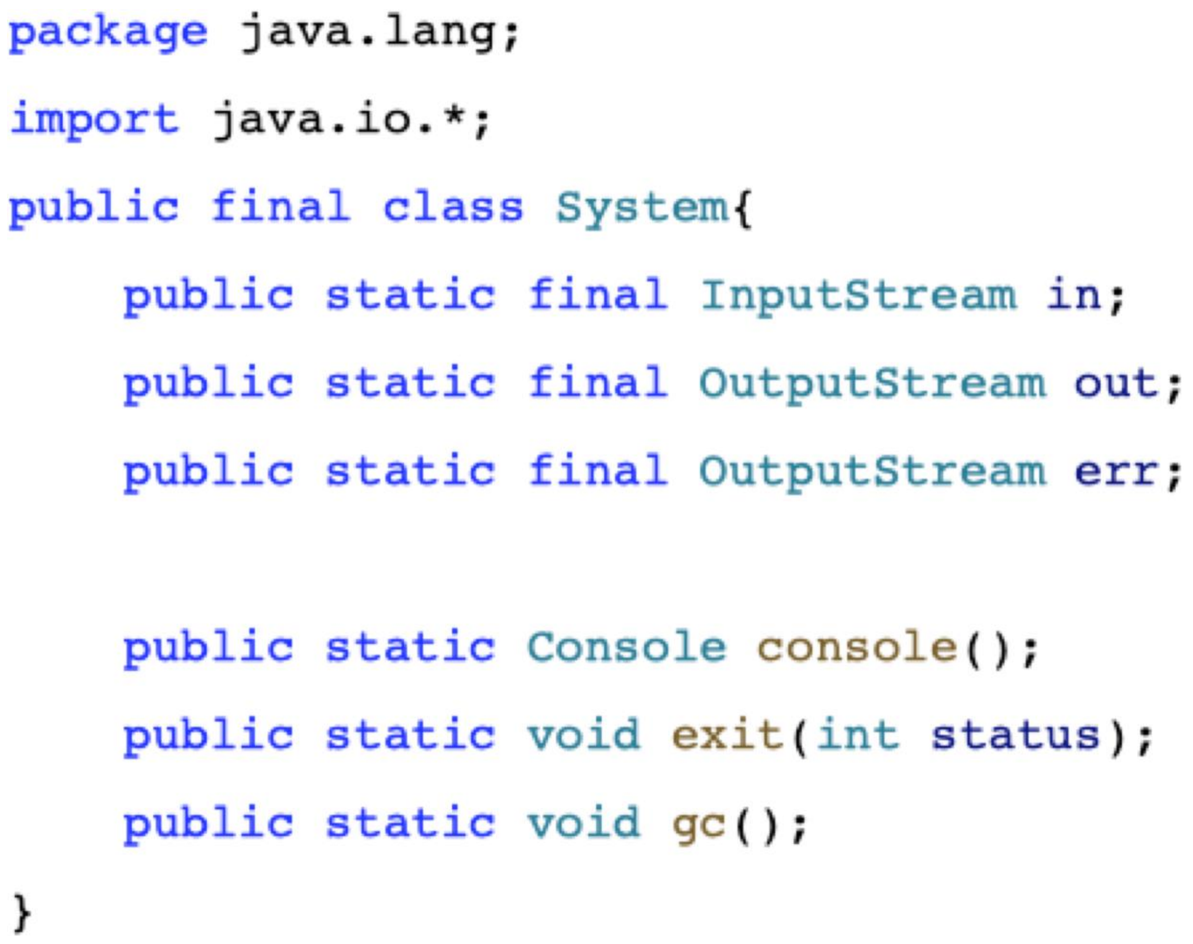
Details Of Selected Type
- Nested Type
- Field
- Constructor
- Method

III

1. java.lang package contains fundamental classes of core Java.

2. It is by default imported in every .java file.

java.lang.System class



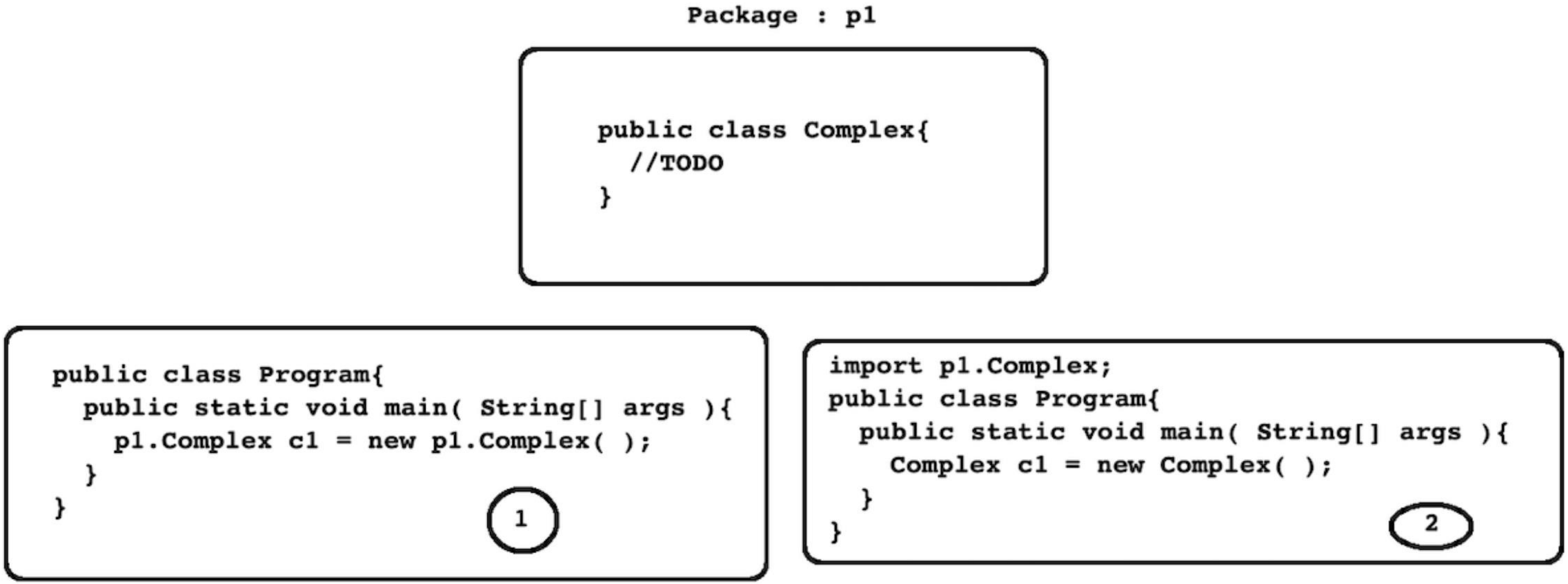
```
package java.lang;
import java.io.*;
public final class System{
    public static final InputStream in;
    public static final OutputStream out;
    public static final OutputStream err;

    public static Console console();
    public static void exit(int status);
    public static void gc();
}
```

Stream

- Stream is an abstraction(object) which either produce(write)/consume(read) information from source to destination.
- Standard stream objects of Java which is associated with console:

  1. **System.in**
     - Ø It represents keyboard.
  2. **System.out**
     - Ø It represents Monitor.
  3. **System.err**
     - Ø Error stream which represents Monitor.

How to access members of package?



Package : p1

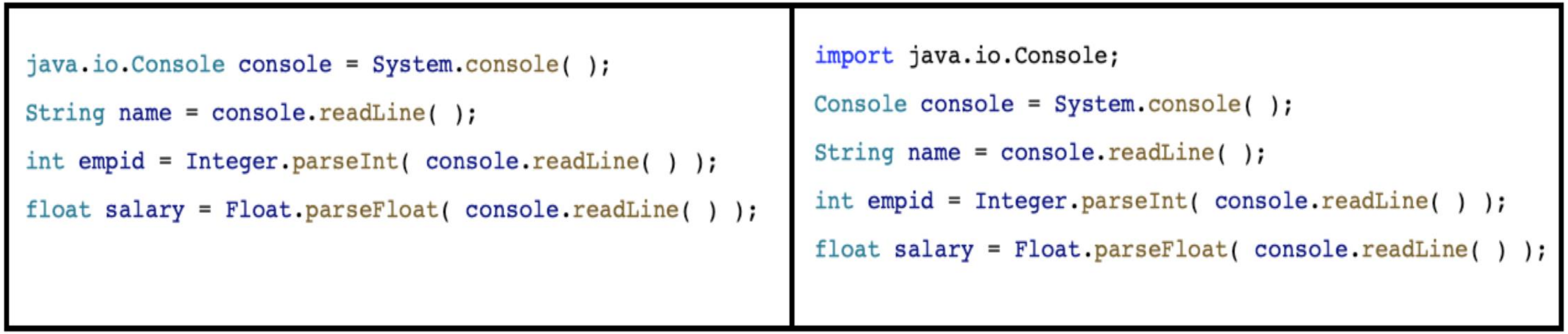
```java
public class Complex{
  //TODO
}
```

```java
public class Program{
  public static void main( String[] args ){
    p1.Complex c1 = new p1.Complex( );
  }
}
```

```java
import p1.Complex;
public class Program{
  public static void main( String[] args ){
    Complex c1 = new Complex( );
  }
}
```

User Input Using Console class

- Console is class declared in java.io package.
- console( ) is a static method of System class which returns reference of java.io.Console class
  - public static Console console( );
- **public String readLine( )** is a method of java.io.Console class.



```
java.io.Console console = System.console( );
String name = console.readLine( );
int empid = Integer.parseInt( console.readLine( ) );
float salary = Float.parseFloat( console.readLine( ) );
```

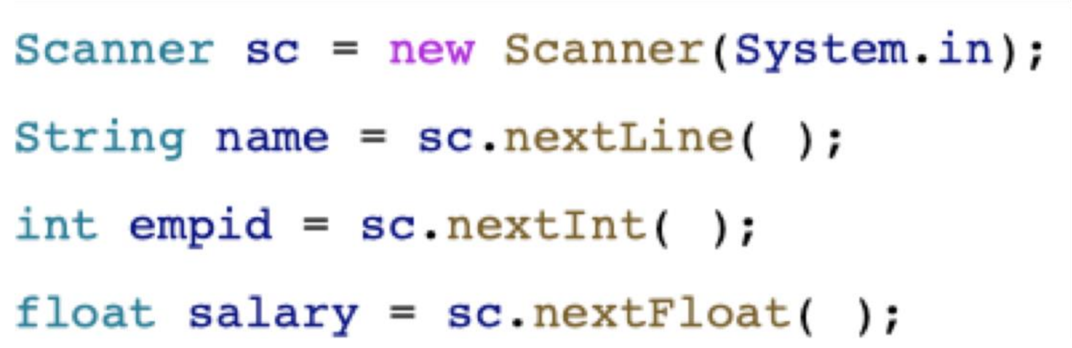
```
import java.io.Console;
Console console = System.console( );
String name = console.readLine( );
int empid = Integer.parseInt( console.readLine( ) );
float salary = Float.parseFloat( console.readLine( ) );
```

User Input Using Scanner class.

- Scanner is a final class declared in java.util package.
- Methods of Scanner class:

```
1. public String nextLine()
2. public int nextInt()
3. public float nextFloat()
4. public double nextDouble()
```

- How to user Scanner?



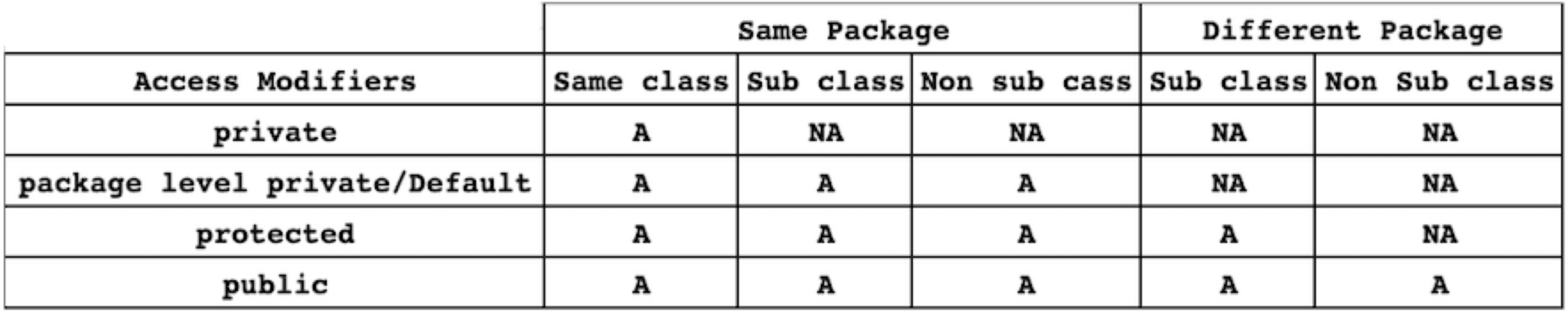
```
Scanner sc = new Scanner(System.in);
String name = sc.nextLine( );
int empid = sc.nextInt( );
float salary = sc.nextFloat( );
```

Modifier

1. ABSTRACT
2. FINAL
3. INTERFACE
4. NATIVE
5. PRIVATE
6. PROTECTED
7. PUBLIC
8. STATIC
9. STRICT
10. SYNCHRONIZED
11. TRANSIENT
12. VOLATILE

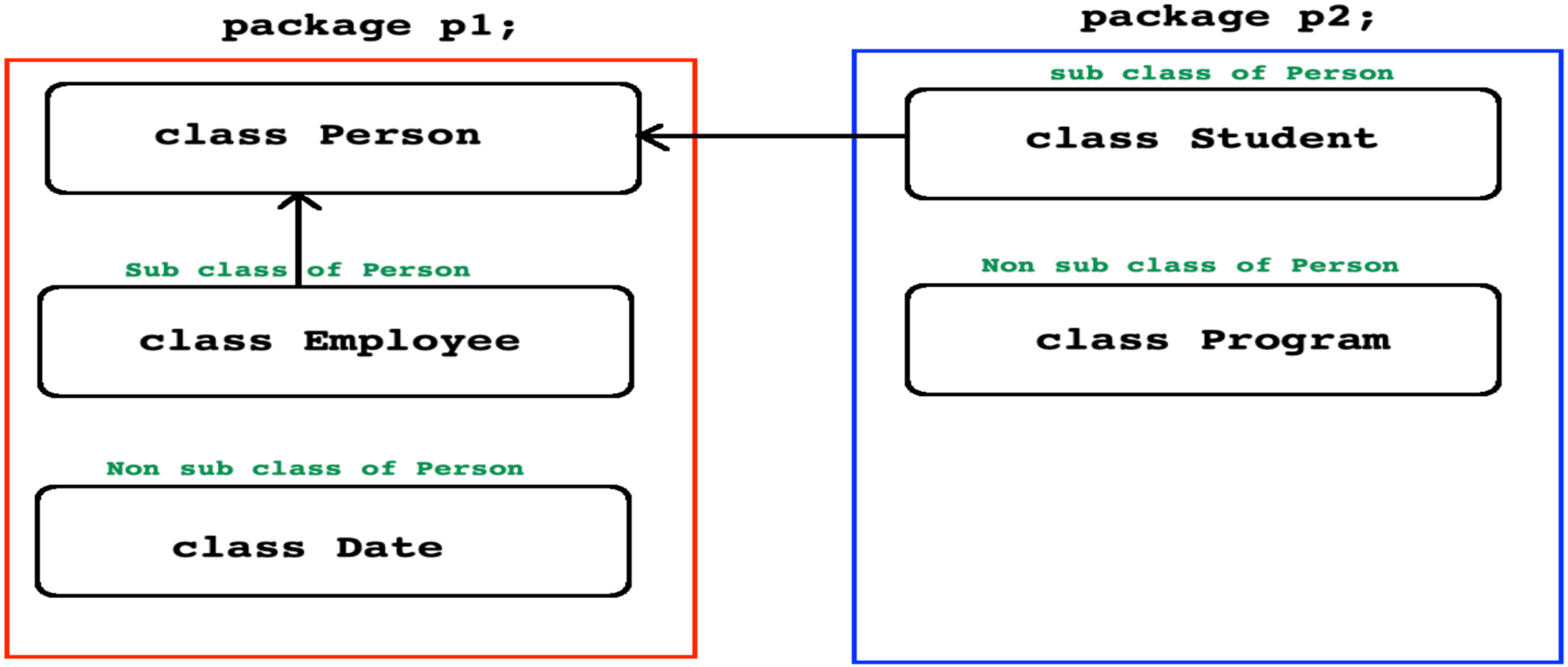
Access Modifier

- If we want to control visibility of members of class then we should use access modifier.
- There are 4 access modifiers in Java:
  1. private
  2. package-level private / default
  3. protected
  4. public



	Same Package			Different Package	
Access Modifiers	Same class	Sub class	Non sub cass	Sub class	Non Sub class
private	A	NA	NA	NA	NA
package level private/Default	A	A	A	NA	NA
protected	A	A	A	A	NA
public	A	A	A	A	A

Access Modifier



Thank you.

akshita.Chanchlani@sunbeaminfo.com