Object Oriented Programming with Java

- Sandeep Kulange

Agenda

- Trainer introduction
- Module information
- Documentation
- Java history
- Version history
- Java platforms
- SDK, JDK, JRE, JVM
- Components of Java platform
- C++ versus Java terminology
- Core Java terminologies
- Structure of Java class
- “Simple Hello” application
- Java application flow of execution
- Comments
- Entry point method
- Data types
- Wrapper class
- Widening and narrowing
- Boxing and unboxing
- Command line arguments

Trainer Introduction

- **Name :** Sandeep Kulange.
- **Designation :** Technical Head
- **Education :**
  - Bachelor of engineering in Information Technology(2005).
  - PG Diploma in Advanced Computing(DAC-Feb 2006).
- **Training Experience**
  - C,C++, Win32 SDK, MS.NET, Core Java, Advanced Java, Database technologies.
- **Email :** sandeepkulange@sunbeaminfo.com

Module Information

- **Name** : Object oriented programming with Java.
- **Duration** : 88 hours
  - Theory : 44 hours
  - Lab : 44 hours
- **Evaluation** : Total 100 Marks
- **Weightage**
  - Theory Exam : 40%
  - Lab Exam : 40%
  - Internal Exam : 20%
- **Text Book**
  - Core and advanced java black book
- **Reference Book**
  - Core Java : Volume 1 and 2 – Cay Horstmann.
  - Java, The complete reference – Herbert Schildt
  - Object-Oriented Analysis and Design with applications – Grady Booch.

Documentation

- **JDK download:**
  - ➢ https://adoptopenjdk.net/
- **Spring Tool Suite 3 download:**
  - ➢ https://github.com/spring-projects/toolsuite-distribution/wiki/Spring-Tool-Suite-3
- **Oracle Tutorial:**
  - ➢ https://docs.oracle.com/javase/tutorial/
- **Java 8 API Documentation:**
  - ➢ https://docs.oracle.com/javase/8/docs/api/
- **MySQL Connector:**
  - ➢ https://downloads.mysql.com/archives/c-j/
- **Core Java Tutorials:**
  1. http://tutorials.jenkov.com/java/index.html
  2. https://www.baeldung.com/java-tutorial
  3. https://www.journaldev.com/7153/core-java-tutorial

Java History

- **James Gosling**, **Mike Sheridan** and **Patrick Naughton** initiated the Java language project in June 1991.

- **Java** was originally **developed by James Gosling** at **Sun Microsystems** and **released in 1995.**

- The language was initially called **Oak** after an oak tree that stood outside Gosling's office.

- Later the project went by the name ***Green*** and was finally renamed ***Java***, from Java coffee, a type of coffee from Indonesia.

- Gosling and his team did a brainstorm session and after the session, they came up with several names such as **JAVA, DNA, SILK, RUBY, etc**.

- Sun Microsystems released the first public implementation as Java 1.0 in 1996.

Java History

- The Java programming language is a general-purpose, concurrent, class-based, object-oriented language.

- The Java programming language is a high-level language.

- The Java programming language is related to C and C++ but is organized rather differently, with a number of aspects of C and C++ omitted and a few ideas from other languages included.

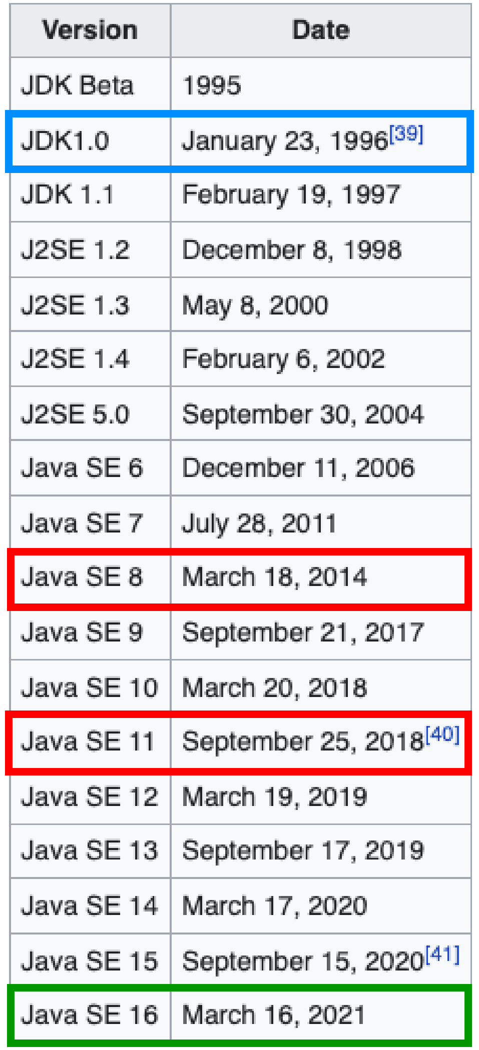
- **It is intended to be a production language, not a research language.**

- The Java programming language is statically typed.

- It promised **Write Once, Run Anywhere** (**WORA**) functionality.

- Standardizing Java Language Specification(JLS) is a job of Java Community Process(**JCP**).

Version History



Version	Date
JDK Beta	1995
JDK1.0	January 23, 1996[39]
JDK 1.1	February 19, 1997
J2SE 1.2	December 8, 1998
J2SE 1.3	May 8, 2000
J2SE 1.4	February 6, 2002
J2SE 5.0	September 30, 2004
Java SE 6	December 11, 2006
Java SE 7	July 28, 2011
Java SE 8	March 18, 2014
Java SE 9	September 21, 2017
Java SE 10	March 20, 2018
Java SE 11	September 25, 2018[40]
Java SE 12	March 19, 2019
Java SE 13	September 17, 2019
Java SE 14	March 17, 2020
Java SE 15	September 15, 2020[41]
Java SE 16	March 16, 2021

- The first version was released on January 23, 1996.

- The acquisition of Sun Microsystems by Oracle Corporation was completed on January 27, 2010

- As of September 2020, Java 8 and 11 are supported as Long Term Support (LTS) versions

- In September 2017, Mark Reinhold, chief Architect of the Java Platform, proposed to change the release train to "one feature release every six months".

- OpenJDK (Open Java Development Kit) is a free and open source implementation of the (Java SE). It is the result of an effort Sun Microsystems began in 2006.

- Java Language and Virtual Machine Specifications: https://docs.oracle.com/javase/specs/

Java Platforms

1. **Java SE**
   - Java Platform Standard Edition.
   - It is also called as Core Java.
   - For general purpose use on Desktop PC’s, servers and similar devices.
2. **Java EE**
   - Java Platform Enterprise Edition.
   - It is also called as advanced Java / enterprise java / web java.
   - Java SE plus various API’s which are useful client-server applications.
3. **Java ME**
   - Java Platform Micro Edition.
   - Specifies several different sets of libraries for devices with limited storage, display, and power capacities.
   - It is often used to develop applications for mobile devices, PDAs, TV set-top boxes and printers.
4. **Java Card**
   - A technology that allows small Java-based applications (applets) to be run securely on smart cards and similar small-memory devices.

SDK, JDK, JRE, JVM

- **SDK** = Development Tools + Documentation + Libraries + Runtime Environment.

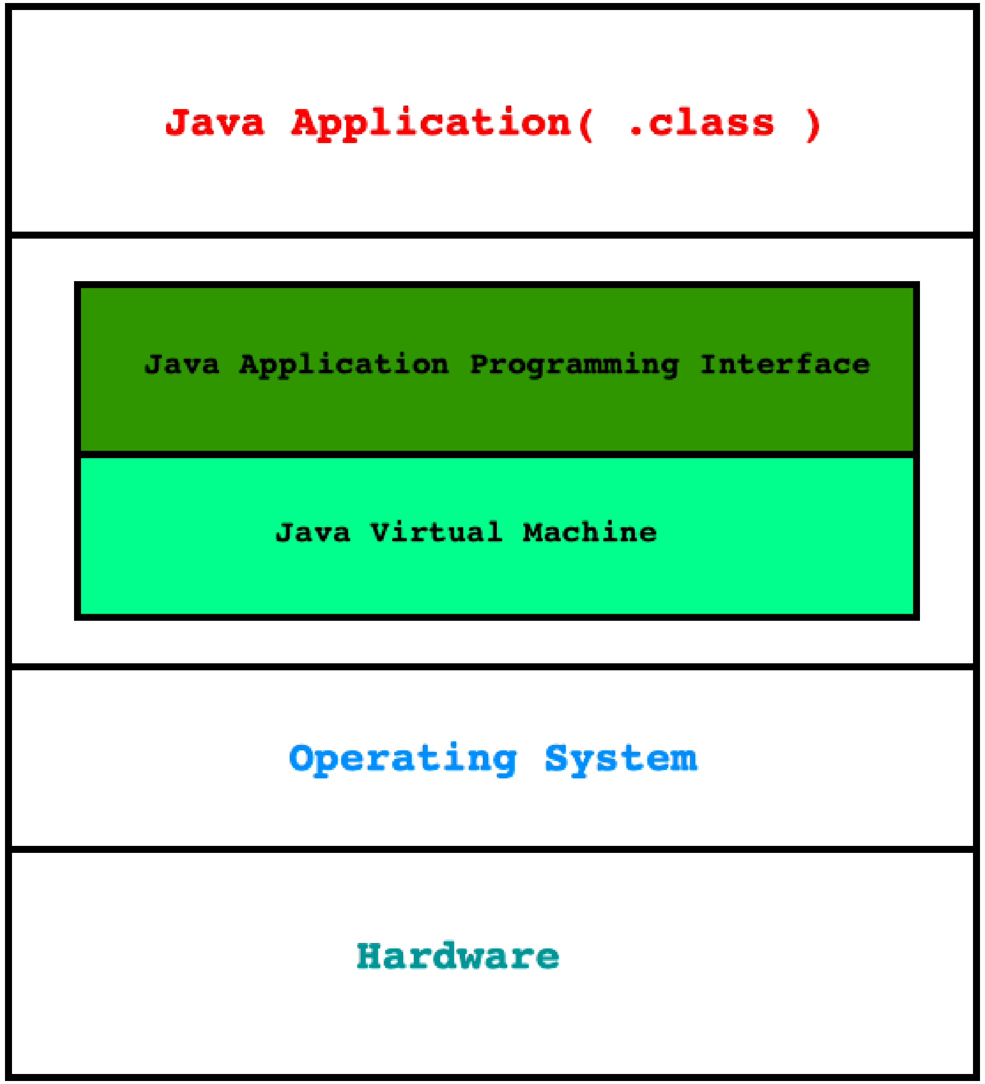
- **JDK** = Java Development Tools + Java Docs + rt.jar + JVM.
  - JDK : Java Development Kit.
  - It is a software, that need to be install on developers machine.
  - We can download it from oracle official website.

- **JDK** = Java Development Tools + Java Docs + **JRE**[rt.jar + JVM].
  - JRE : Java Runtime Environment.
  - rt.jar and JVM are integrated part of JRE.
  - JRE is a software which comes with JDK. We can also download it separately.
  - To deploy application, we should install it on client's machine.

- rt.jar file contains core Java API in compiled form.

- **JVM** : An engine, which manages execution of Java application.

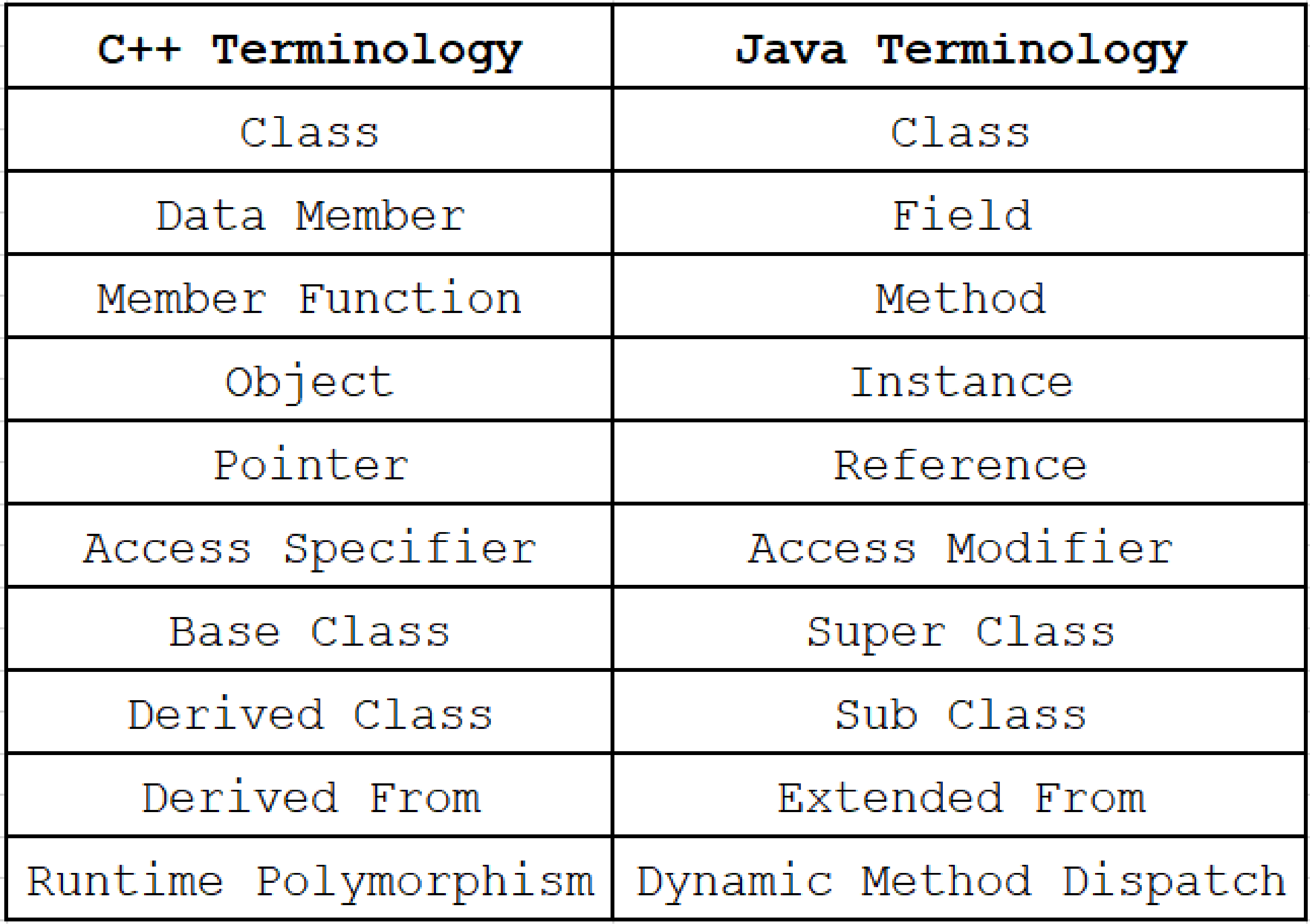
Components of the Java platform


Java Application(.class)
Java Application Programming Interface
Java Virtual Machine
Operating System
Hardware

Java Platform
[Software Platform]

[Hardware Based Platform]

C++ versus Java terminology



C++ Terminology	Java Terminology
Class	Class
Data Member	Field
Member Function	Method
Object	Instance
Pointer	Reference
Access Specifier	Access Modifier
Base Class	Super Class
Derived Class	Sub Class
Derived From	Extended From
Runtime Polymorphism	Dynamic Method Dispatch

Core Java terminologies

- A method that we can not redefine/override it inside sub class is called **final method.**

- A class, that we can not extend is called **final class.**

- A method of a class, which do not have body is called **abstract method.**

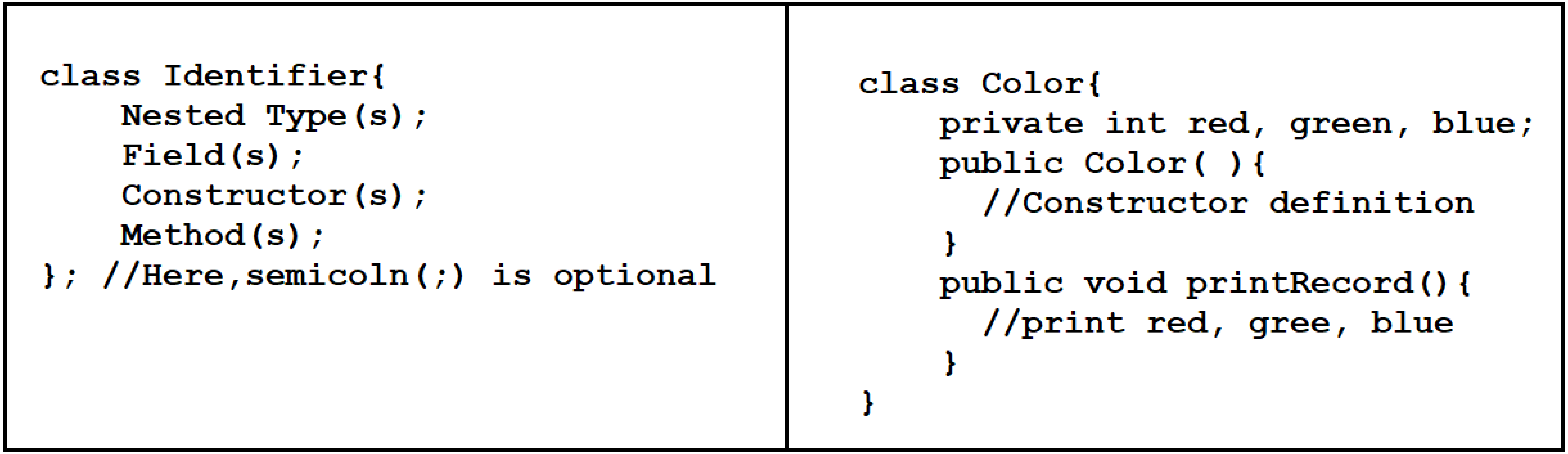
- A class from which, we can not create object is called **abstract class.**

- A method of a class, which is having body is called **concrete method.**

- A class from which, we can create object is called **concrete class.**

- **[Note:** All above topics will be covered in detail in inheritance.]

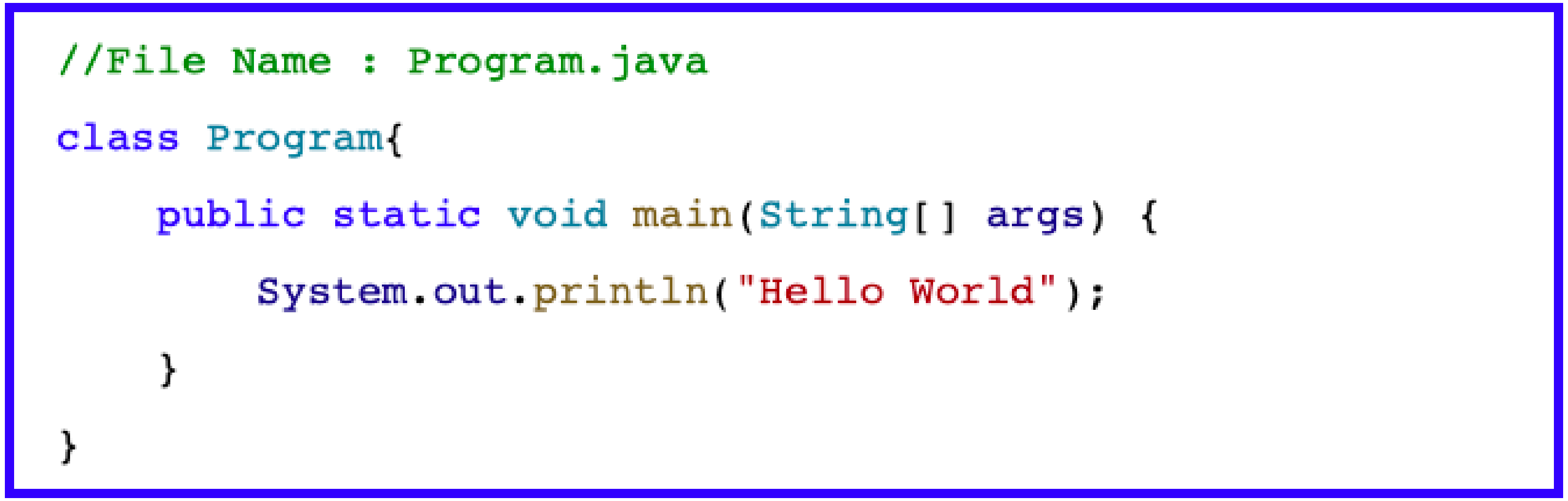
Structure of Java class



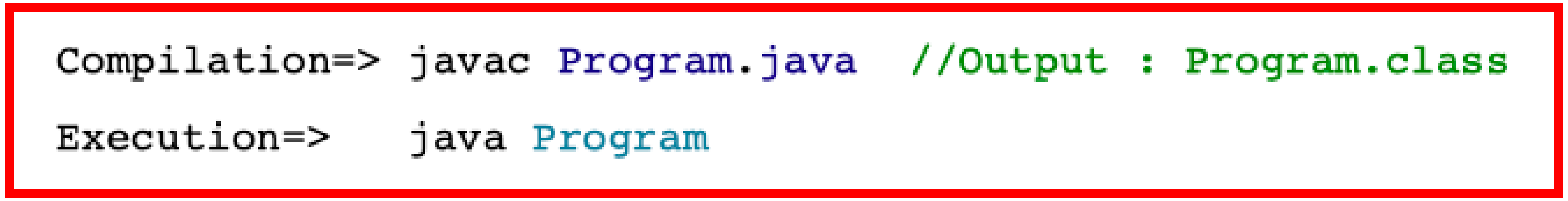
```
class Identifier{
    Nested Type(s);
    Field(s);
    Constructor(s);
    Method(s);
}; //Here,semicoln(;) is optional
```

```
class Color{
    private int red, green, blue;
    public Color( ){
      //Constructor definition
    }
    public void printRecord(){
      //print red, gree, blue
    }
}
```

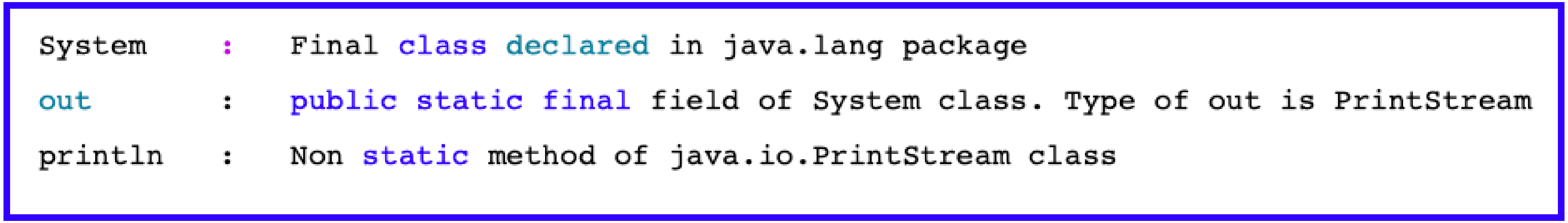
Simple Hello Application



```
//File Name : Program.java
class Program{
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

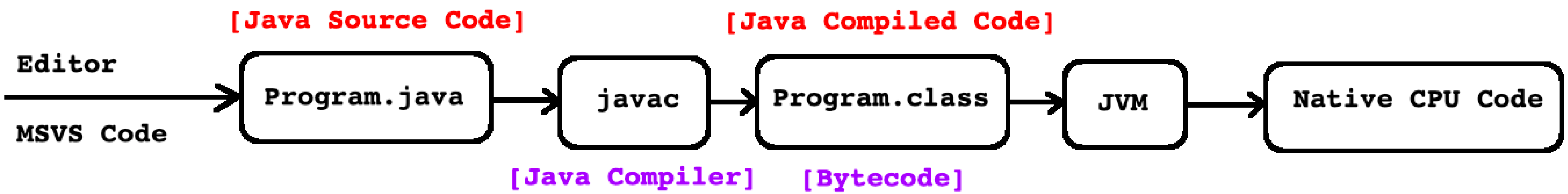


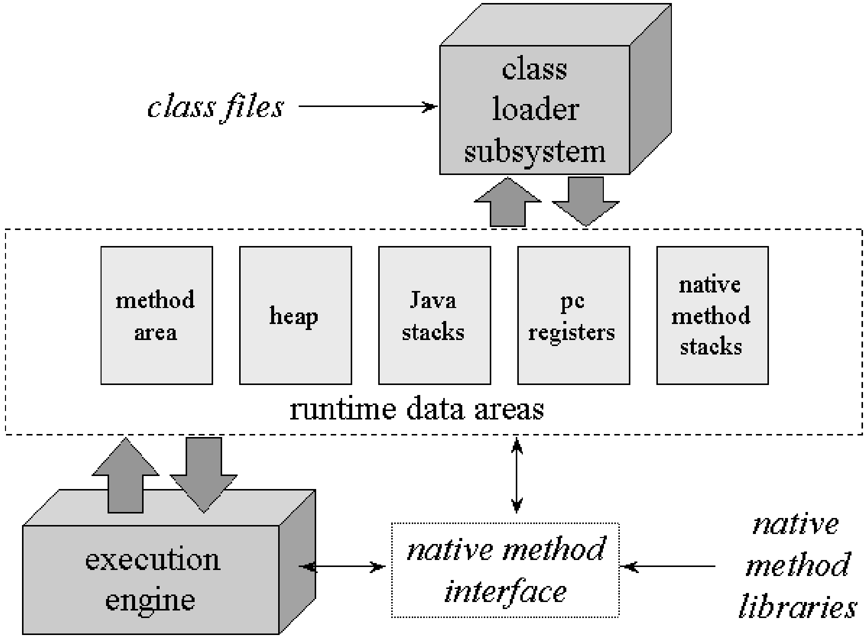
```
Compilation=> javac Program.java  //Output : Program.class
Execution=>   java Program
```



```
System    :   Final class declared in java.lang package
out       :   public static final field of System class. Type of out is PrintStream
println   :   Non static method of java.io.PrintStream class
```

Java application flow of execution





[JVM Architecture]

Comments

- Comments should be used to give overviews of code and provide additional information that is not readily available in the code itself. Comments should contain only information that is relevant to reading and understanding the program.
- Types of Comments:
  - **Implementation Comment**
    - Implementation comments are those found in C++, which are delimited by /*...*/, and //.
      1. Single-Line Comment
      2. Block Comment(also called as multiline comment)
  - **Documentation Comment**
    - Documentation comments (known as "doc comments") are Java-only, and are delimited by **/**...*/**.
    - Doc comments can be extracted to HTML files using the javadoc tool.
- Ref: https://www.oracle.com/java/technologies/javase/codeconventions-comments.html

Entry point method

- Syntax:
  1. **public static void main(String[] args)**
  2. **public static void main(String... args)**

- Java compiler do not check/invoke main method. JVM invoke main method.

- When we start execution of Java application then JVM starts execution of two threads:
  1. Main thread : responsible for invoking main method.
  2. Garbage Collector : responsible for deallocating memory of unused object.

- We can overload main method in Java.

- We can define main method per class. But only one main method can be considered as entry point method.

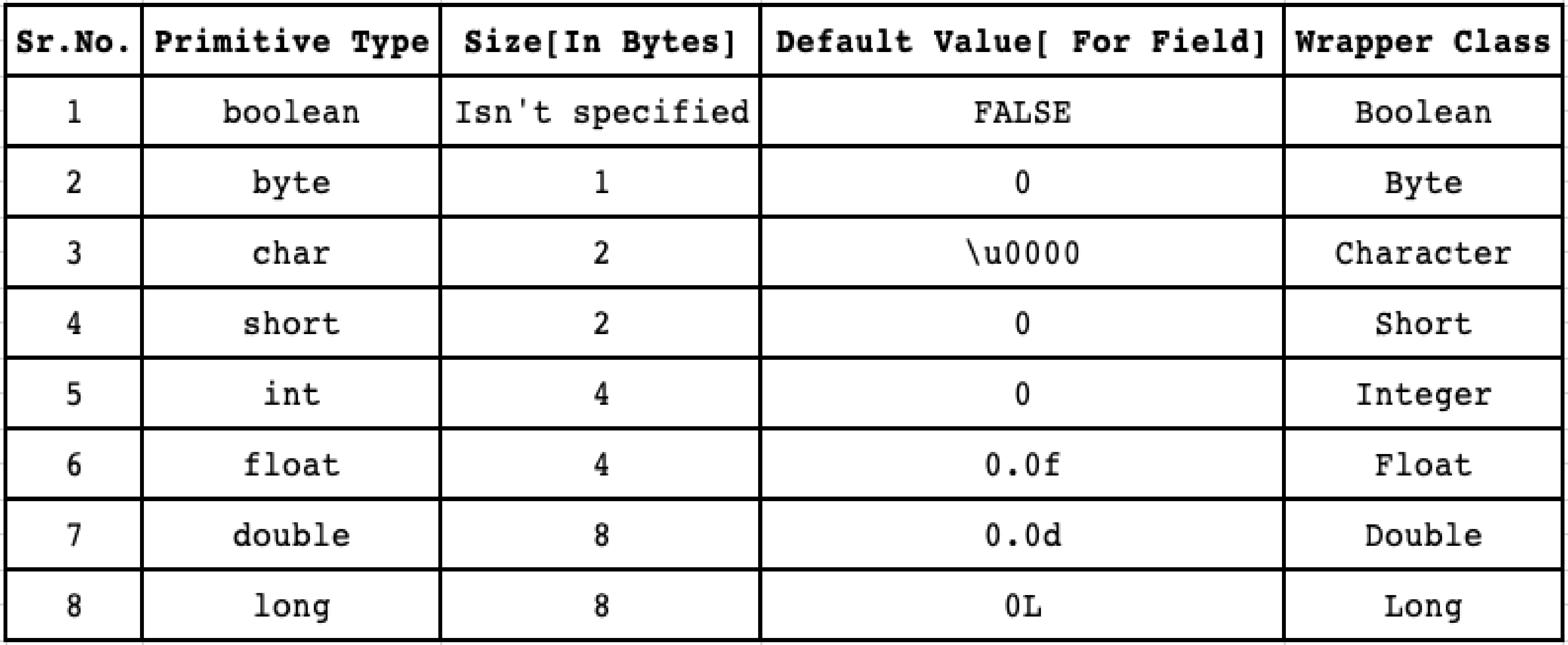
Data Types

- Data type of any variable decide following things:
  1. **Memory:** How much memory is required to store the data.
  2. **Nature:** Which kind of data is allowed to store inside memory.
  3. **Operation:** Which operations are allowed to perform on the data stored in memory.
  4. **Range:** Set of values that we can store inside memory.

- The Java programming language is a statically typed language, which means that every variable and every expression has a type that is known at compile time.
  - Types of data type:
    1. **Primitive type(also called as value type)**
       - **boolean** type
       - Numeric type
         1. Integral types(**byte**, **char**, **short**, **int**, **long**)
         2. Floating point types(**float**, **double**
    2. **Non primitive type(also called as reference type)**
       - **Interface, Class, Type variable, Array**

- Ref: https://docs.oracle.com/javase/specs/jls/se8/html/jls-4.html

Data Types



Sr.No.	Primitive Type	Size[In Bytes]	Default Value[For Field]	Wrapper Class
1	boolean	Isn't specified	FALSE	Boolean
2	byte	1	0	Byte
3	char	2	\u0000	Character
4	short	2	0	Short
5	int	4	0	Integer
6	float	4	0.0f	Float
7	double	8	0.0d	Double
8	long	8	0L	Long

Wrapper class

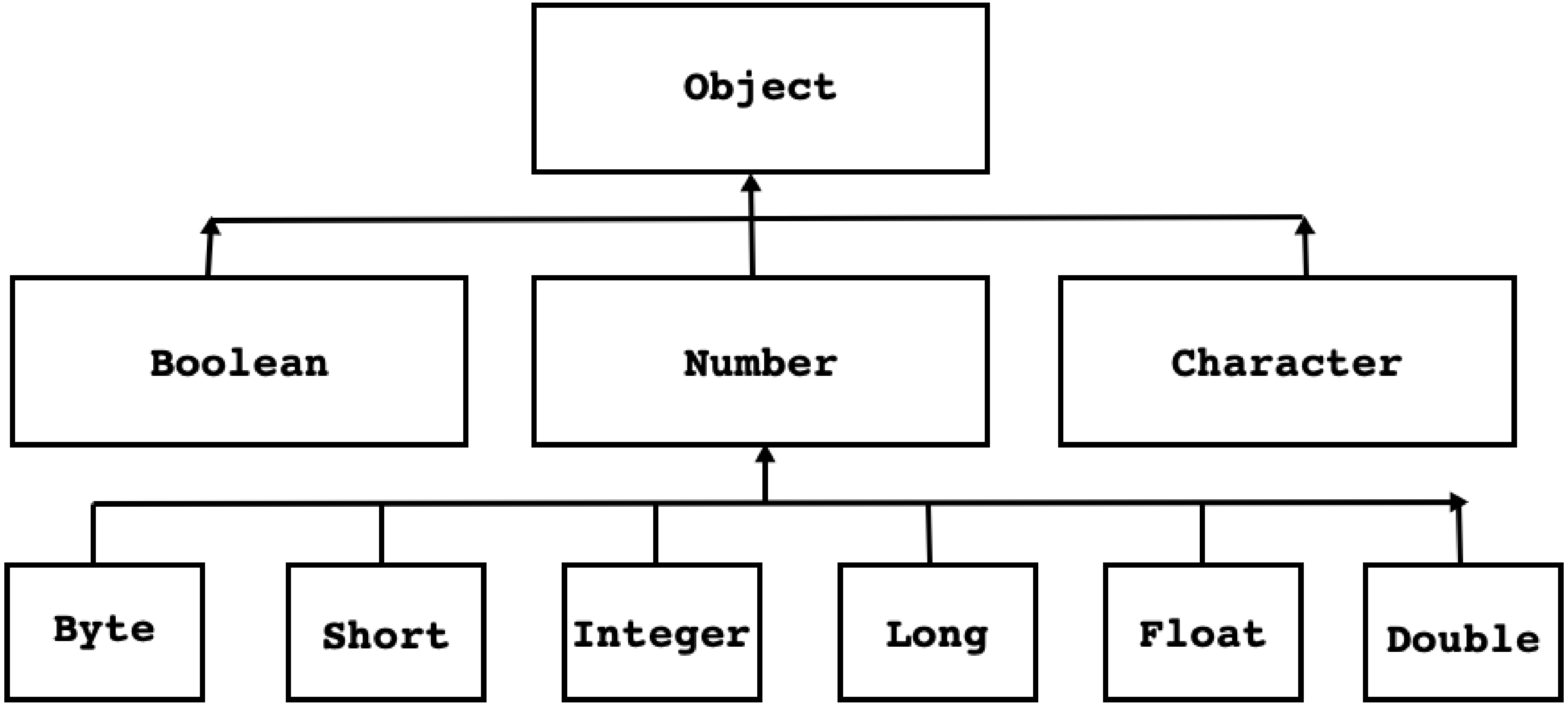
- In Java, primitive types are not classes. But for every primitive type, Java has defined a class. It is called wrapper class.

- All wrapper classes are final.

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

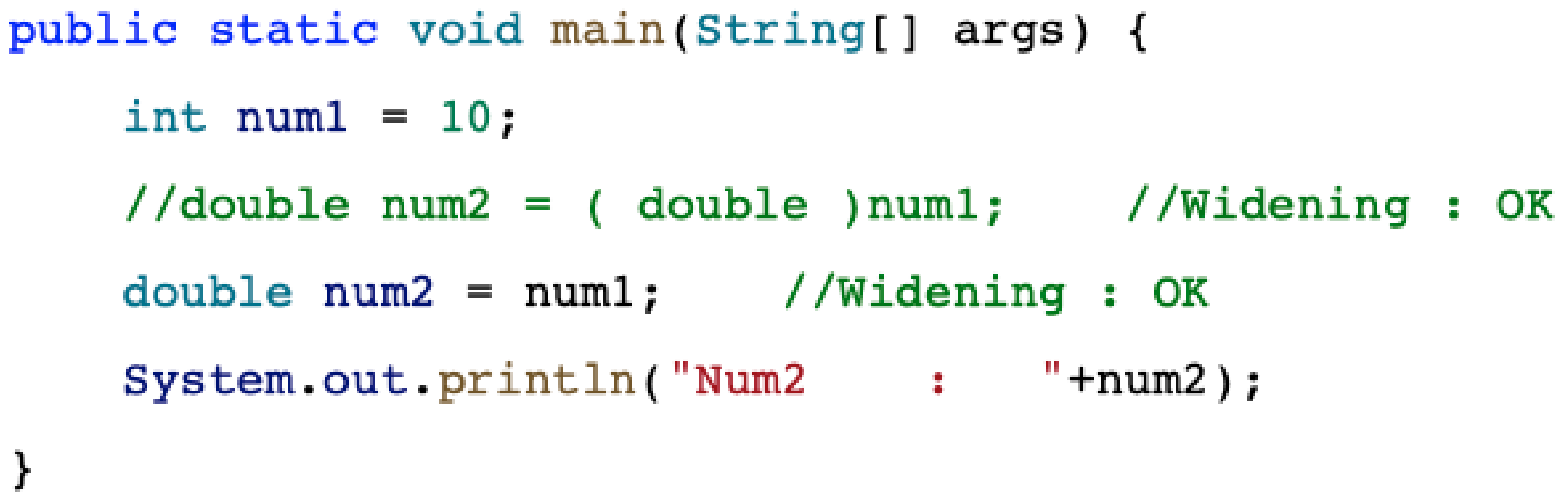
- Uses of Wrapper class
    1. To parse string(i.e. to convert state of string into numeric type).
    2. To store value of primitive type into instance of generic class, type argument must be wrapper class.
        - ➢ **Stack<int> stk = new Stack<int>( );   //Not OK**
        - ➢ **Stack<Integer> stk = new Stack<Integer>( );   //OK**

Wrapper class



Widening

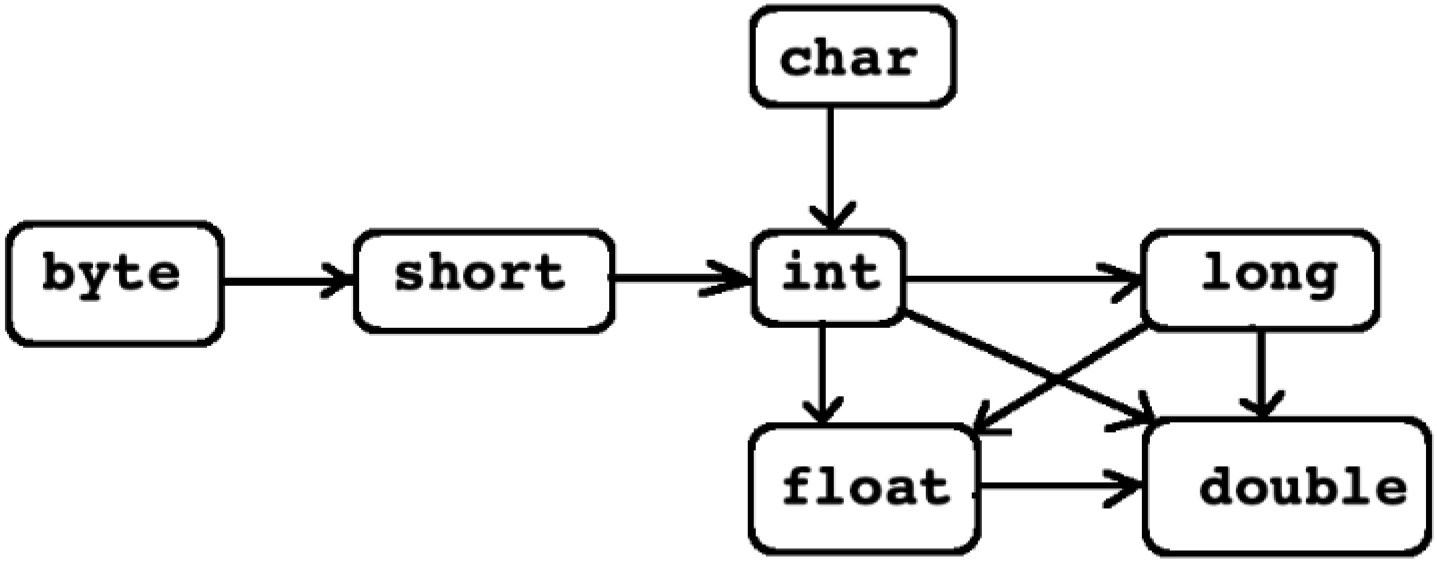
- Process of converting value of variable of narrower type into wider type is called widening.



```
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, explicit type casting is optional.

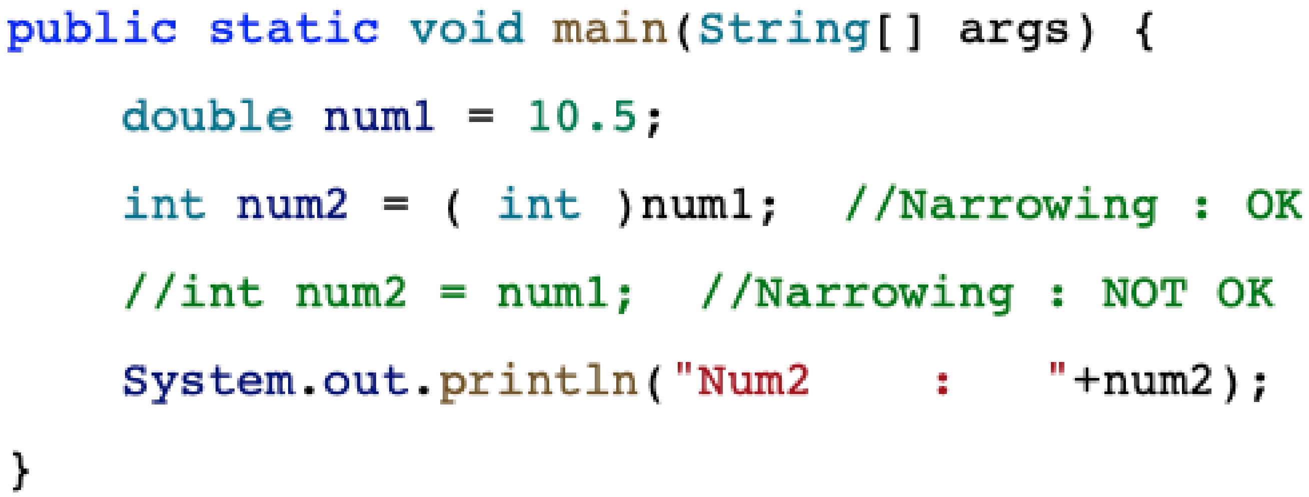
Widening



Widening Conversion

Narrowing

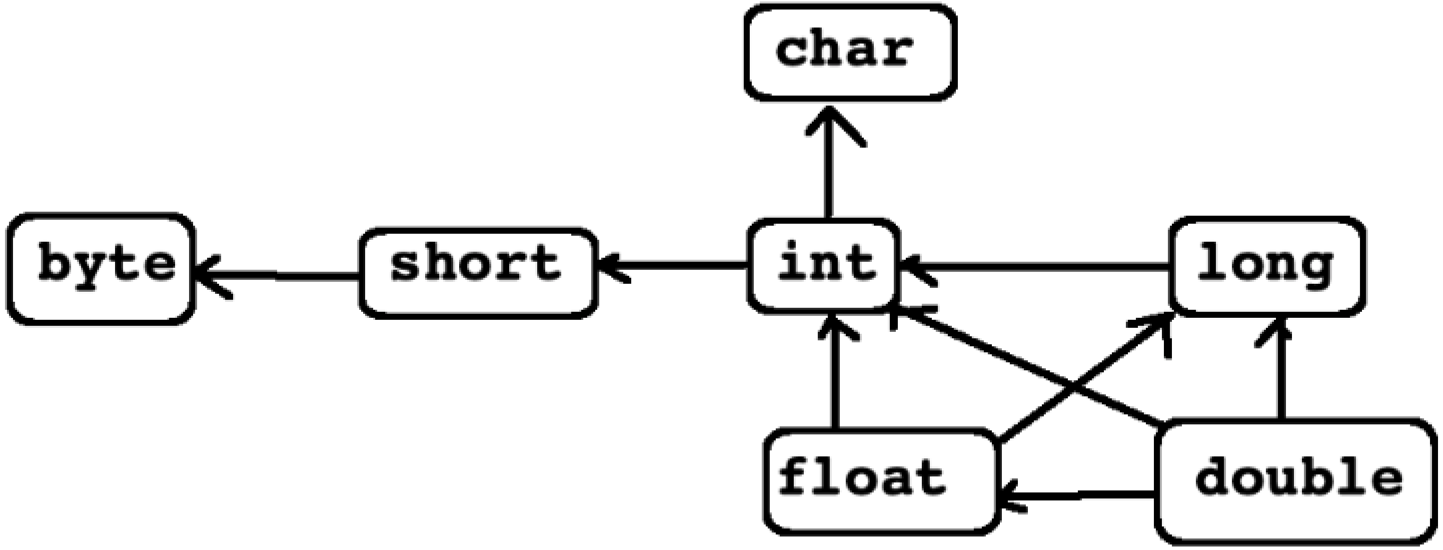
- Process of converting value of variable of wider type into narrower type is called narrowing.



```
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.

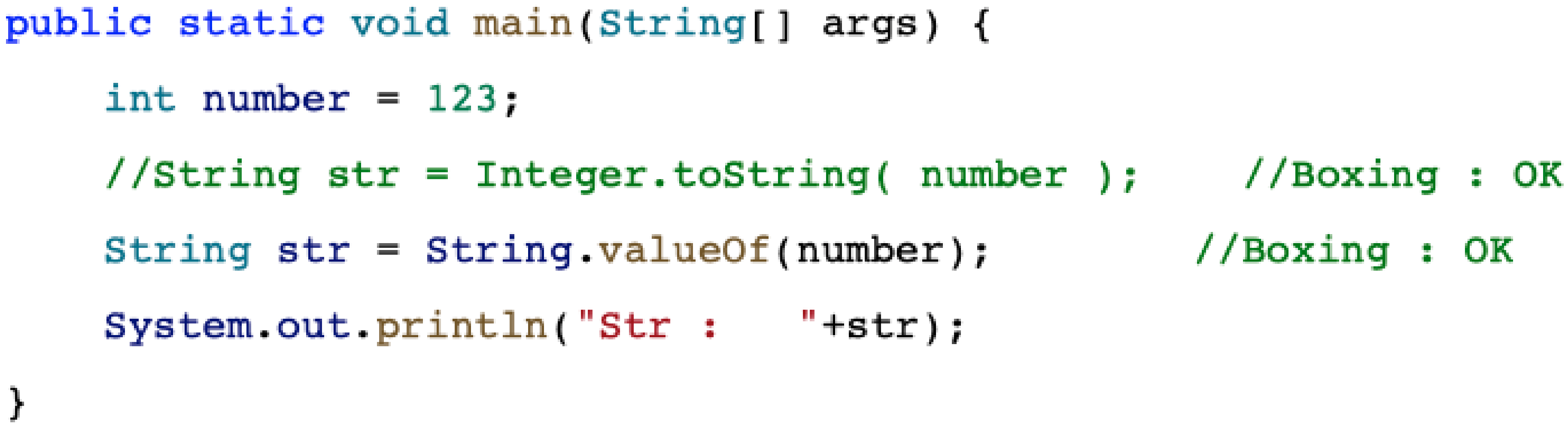
Narrowing



Narrowing Conversion.

Boxing

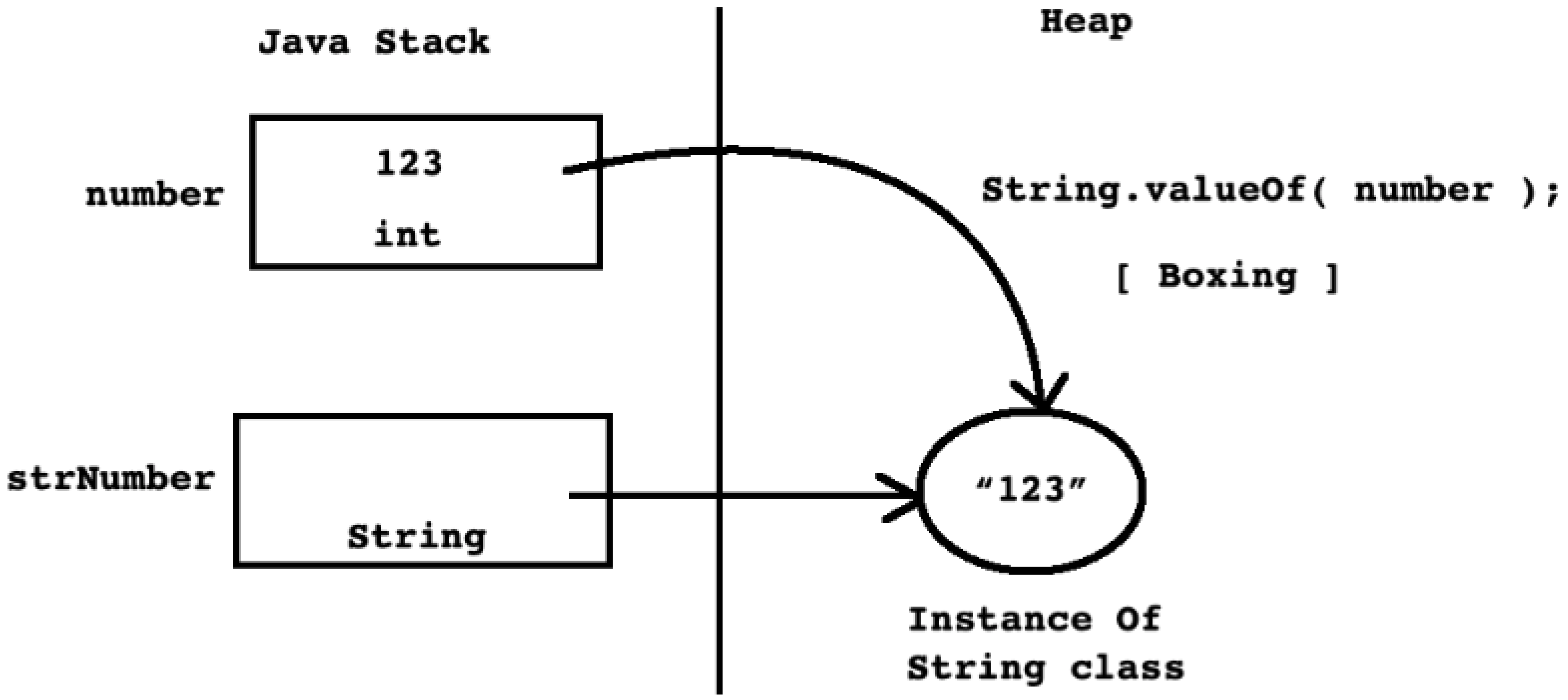
- Process of converting value of variable of primitive type into not primitive type is called **boxing**.



```
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);
}
```

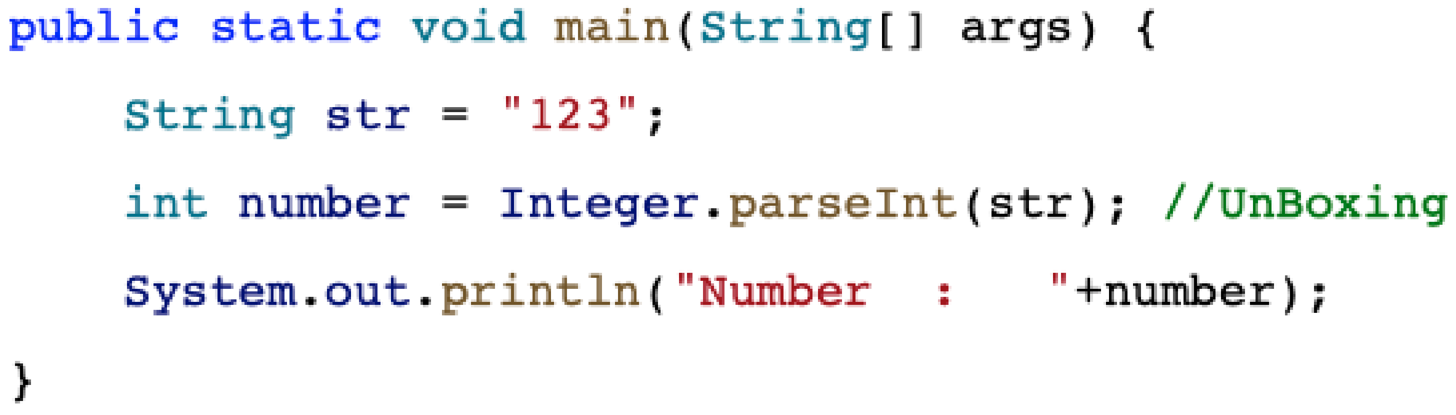
Boxing

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



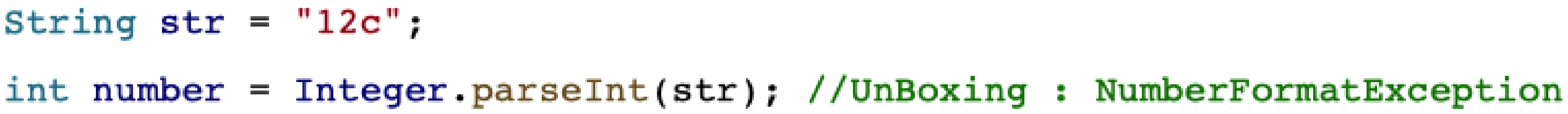
Unboxing

- 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);
}
```

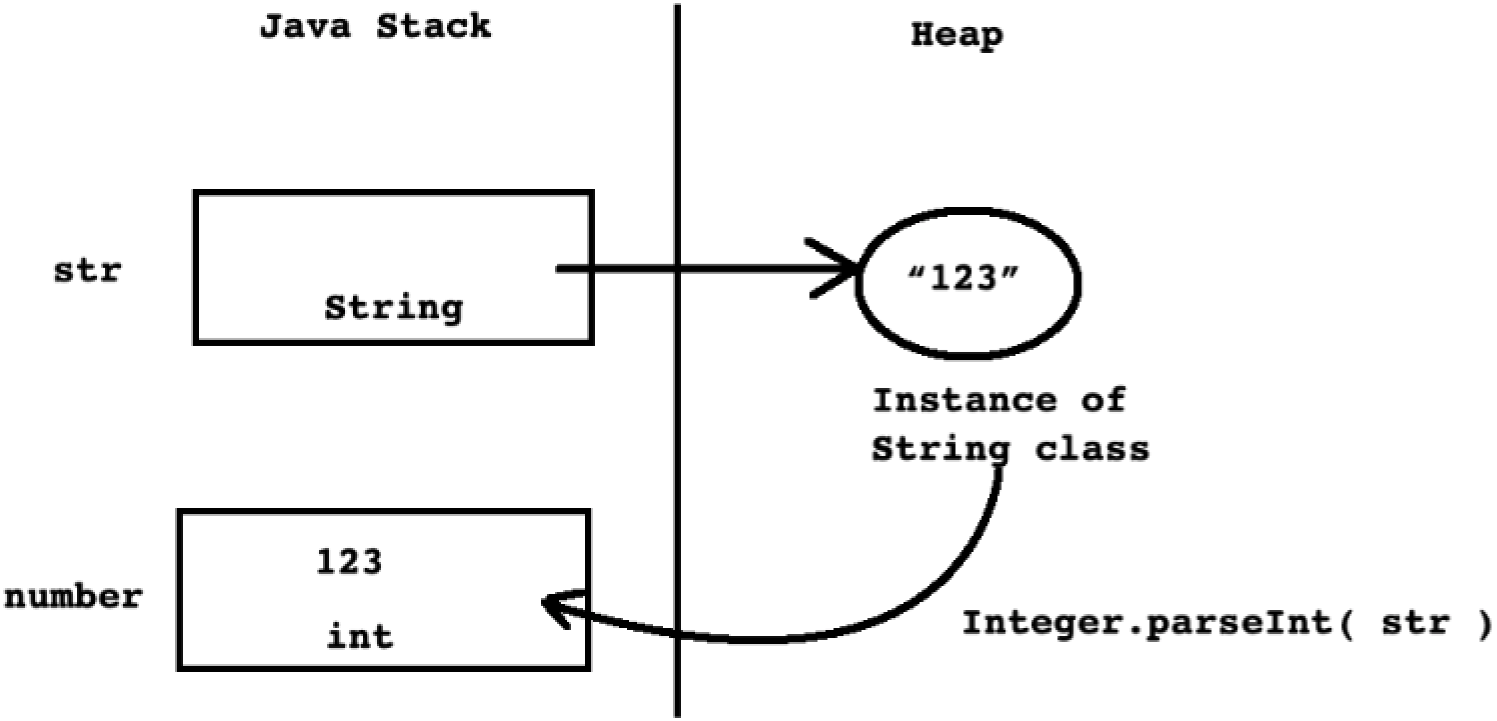
- If string does not contain parseable numeric value then **parseXXX( )** method throws **NumberFormatException**.



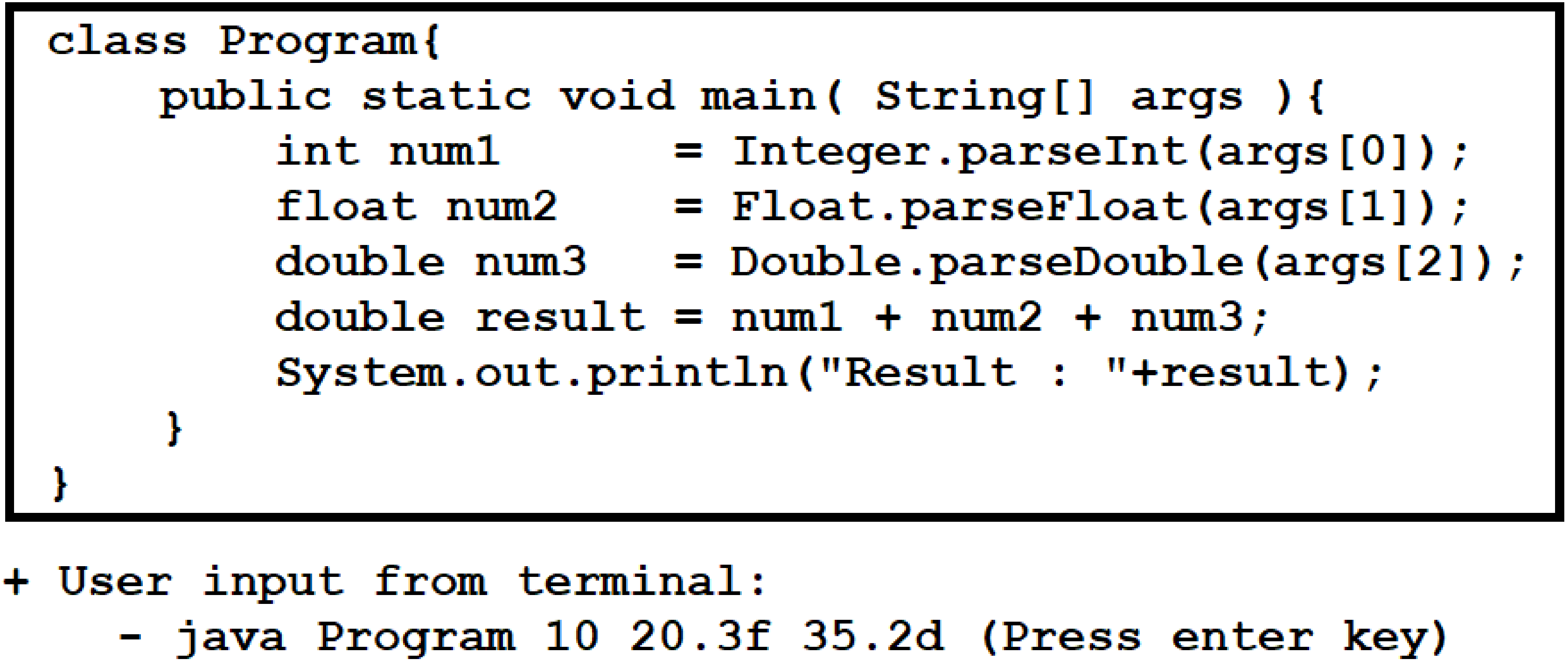
```
String str = "12c";
int number = Integer.parseInt(str); //UnBoxing : NumberFormatException
```

Unboxing

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



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)

Thank you.

[sandeepkulange@sunbeaminfo.com]