Object Oriented Programming with Java 8
PG-DAC

Rohan Paramane

Agenda

- Class
- Object
- Wrapper Class
- Widening
- Narrowing
- Boxing
- UnBoxing
- Command Line Arguments

Class

- Consider following examples:
  1. day, month, year - related to – Date
  2. hour, minute, second - related to – Time
  3. red, green, blue - related to Color
  4. real, imag - related to – Complex
  5. xPosition, yPosition - related to Point
  6. number, type, balance - related to Account
  7. name, id, salary - related to Employee

- If we want to group related data elements together then we should use/define class in Java.

```
class Date{
    int day;        //Field
    int month;      //Field
    int year;       //Field
}
```

```
class Employee{
    String name;    //Field
    int id;         //Field
    float salary;   //Field
}
```

Class and object

- class is a non primitive/reference type in Java.
- Classes and Objects are basic concepts of Object Oriented Programming which revolve around the real life entities.

A **class** is a user defined blueprint or prototype or template : from which objects are created.
It represents the set of properties(DATA) and methods(ACTIONs) that are common to all objects of one type.

Class declaration includes

1. Access specifiers : A class can be public or has default access
2. Class name: The name should begin with a capital letter & then follow camel case convention
3. Superclass(if any): The name of the class's parent (superclass), if any, preceded by the keyword extends. (Implicit super class of all java classes is java.lang.Object)
4. Interfaces(if any): A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
   eg : public class Emp extends Person implements Runnable,Comparable{...}
5. Body: The class body surrounded by braces, { }.
6. Constructors are used for initializing state of the new object/s.
7. Fields are variables that provides the state of the class and its objects
8. Methods are used to implement the behavior of the class and its objects.

eg : Student,Employee,Flight,PurchaseOrder, Shape ,BankAccount.....

Object

- It is a basic unit of Object Oriented Programming and represents the real life entities. A typical Java program creates many objects, which interact by invoking methods.

- An object consists of :

  - State : It is represented by attributes of an object. (properties of an object) / instance variables(non static)
  - Behavior : It is represented by methods of an object (actions upon data)
  - Identity : It gives a unique identity to an object and enables one object to interact with other objects. eg : Emp id / Student PRN / Invoice No

- Creating an object
  - The new operator instantiates a class by allocating memory for a new object and returning a reference to that memory. The new operator also invokes the class constructor.

Class

- Field
  - Ø A variable declared inside class / class scope is called a field.
  - Ø Field is also called as attribute or property.
- Method
  - Ø A function implemented inside class/class scope is called as method.
  - Ø Method is also called as operation, behavior or message.
- Class
  - Ø Class is a collection of fields and methods.
  - Ø Class can contain
    1. Nested Type
    2. Field
    3. Constructor
    4. Method
- **Instance** : In Java, Object is also called as instance.

Note: All stand-alone C++ programs require a function named main and can have numerous other functions. Java does not have stand alone functions, all functions (called methods) are members of a class. All classes in Java ultimately inherit from the Object class, while it is possible to create inheritance trees that are completely unrelated to one another in C++. In this sense , Java is a pure Object oriented language, while C++ is a mixture of Object oriented and structure language.

Instantiation

- Process of creating instance/object from a class is called as instantiation.

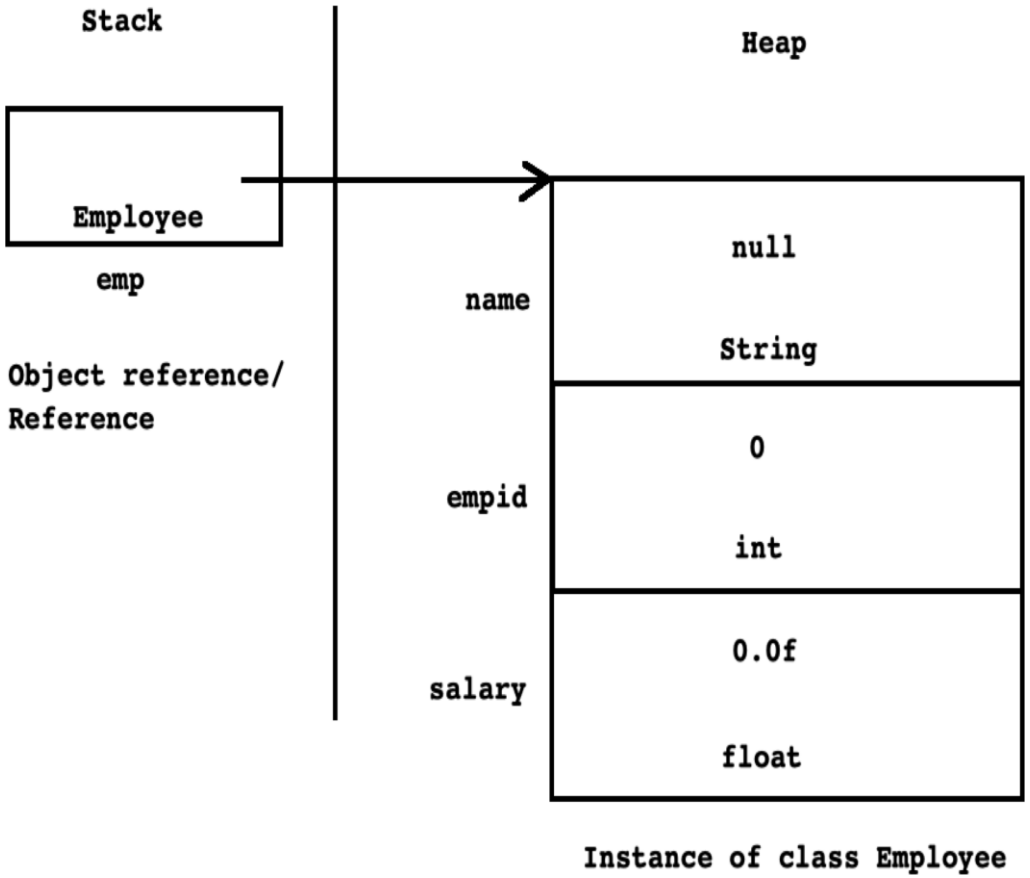
- In C programming language
  - Ø Syntax : struct StructureName identifier_name; struct
  - Ø Employee emp;

- In C++ programming language
  - Ø Syntax : [class] ClassName identifier_name;
  - Ø Employee emp;

- In Java programming language
  - Ø Syntax : ClassName identifier_name = new ClassName( );
  - Ø Employee emp = new Employee();

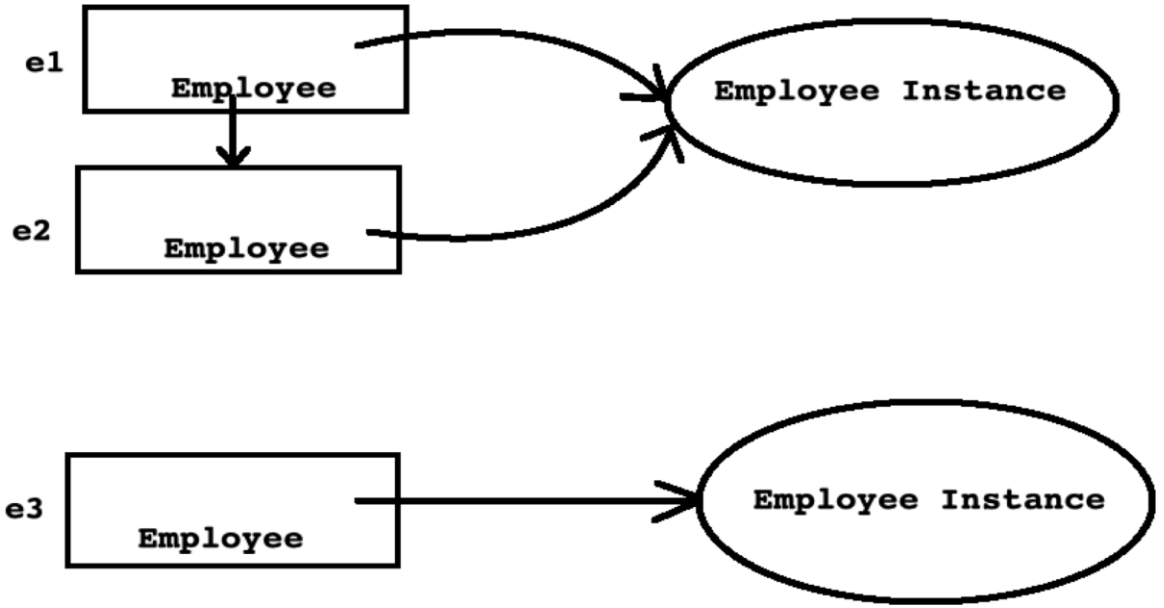
- **Every instance on heap section is anonymous.**

Instantiation



For eg :

```
1. Employee e1 = new Employee();
2. Employee e2 = e1;
3. Employee e3 = new Employee();
```



Wrapper class

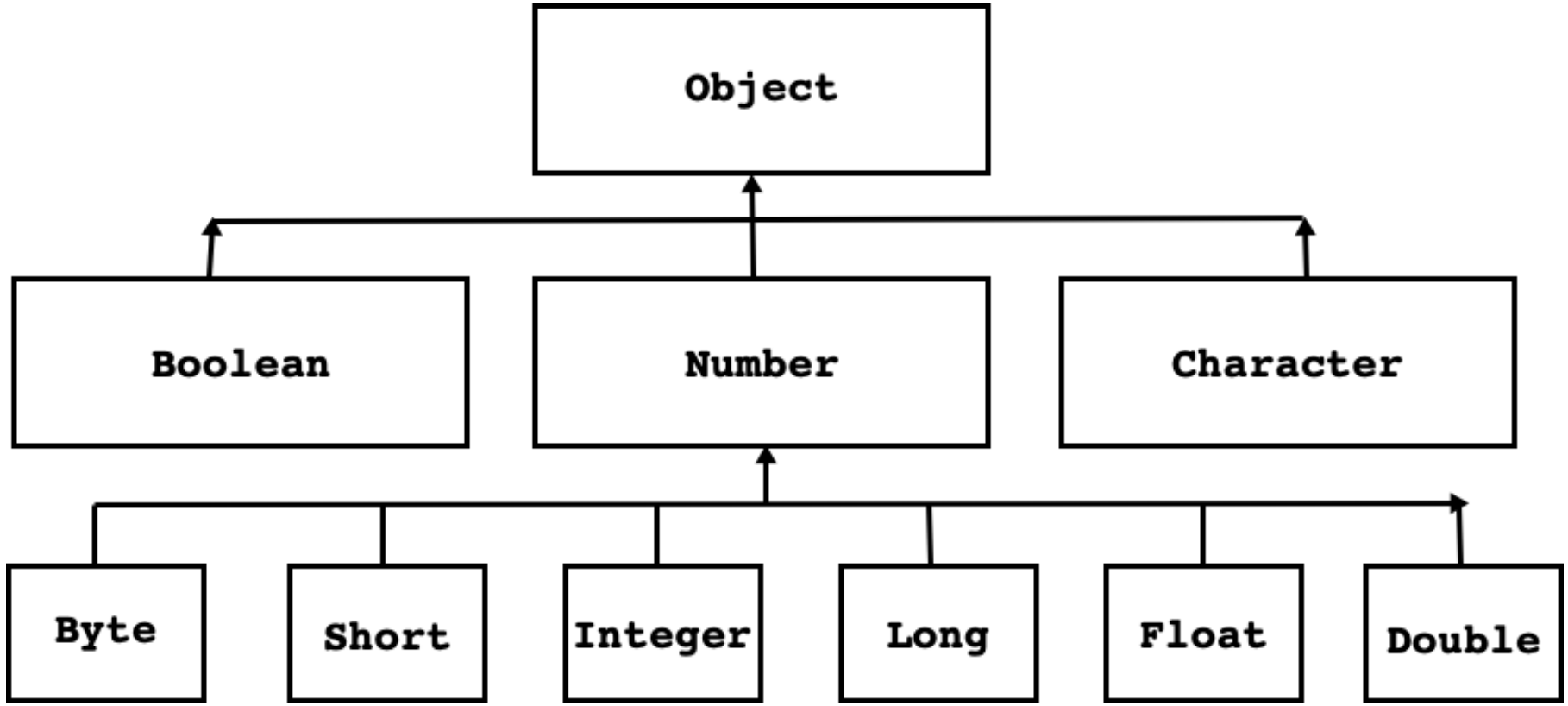
- In Java, primitive types are not classes. But for every primitive type, Java has defined a class. It is called wrapper class.

- 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



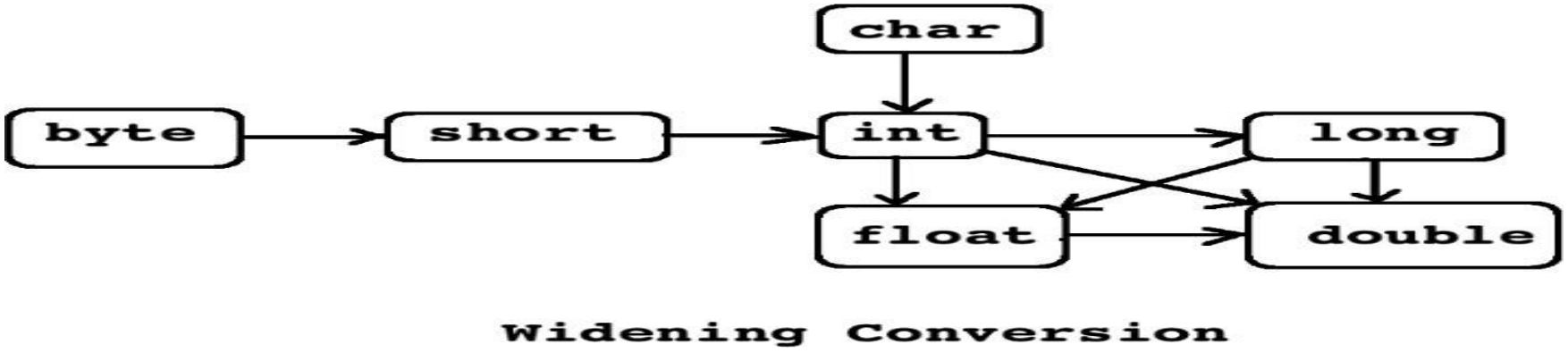
Widening

- 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

The range of values that can be represented by a float or double is much larger than the range that can be represented by a long. Although one might lose significant digits when converting from a long to a float, it is still a "widening" operation because the range is wider.

A widening conversion of an int or a long value to float, or of a long value to double, may result in loss of precision - that is, the result may lose some of the least significant bits of the value. In this case, the resulting floating-point value will be a correctly rounded version of the integer value, using IEEE 754 round-to-nearest mode.
Note that a double can exactly represent every possible int value.

long --->float ---is considered automatic type of conversion(since float data type can hold larger range of values than long data type)

Rules

- src & dest - must be compatible, typically dest data type must be able to store larger magnitude of values than that of src data type.

- 1. Any arithmetic operation involving byte,short --- automatically promoted to --int
- 2. int & long ---> long
- 3. long & float ---> float
- 4. byte,short.....& float & double----> double

Narrowing (Forced Conversion)

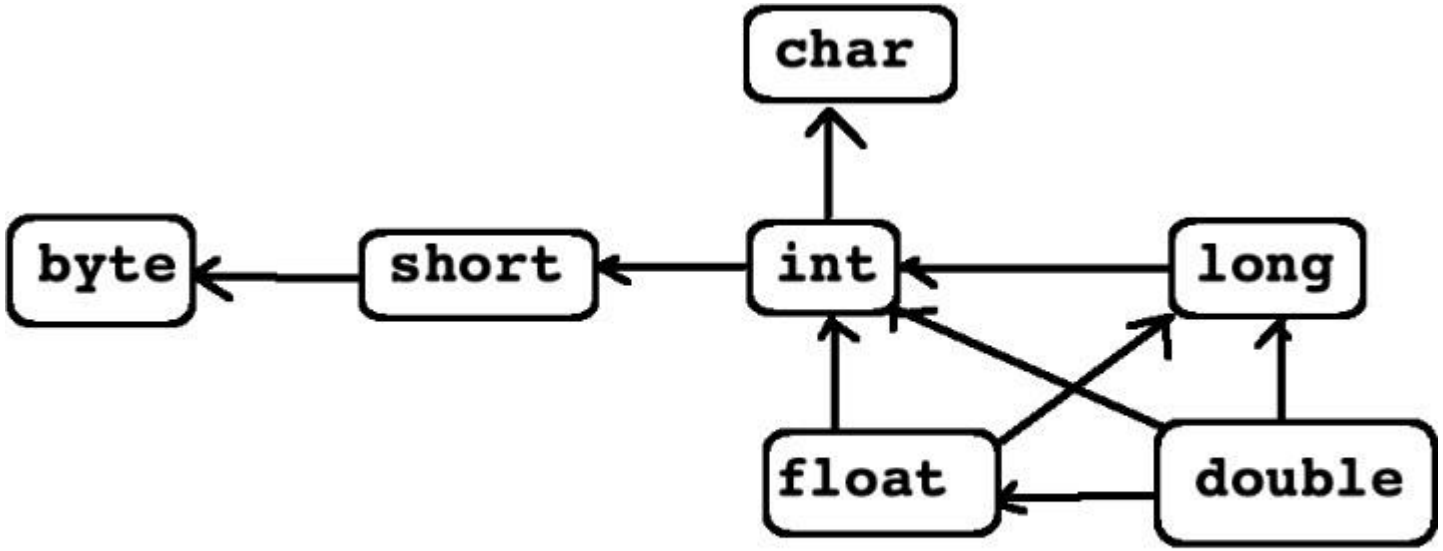
- Process of converting value of variable of wider type into narrower type is called narrowing.

```
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.

eg ---

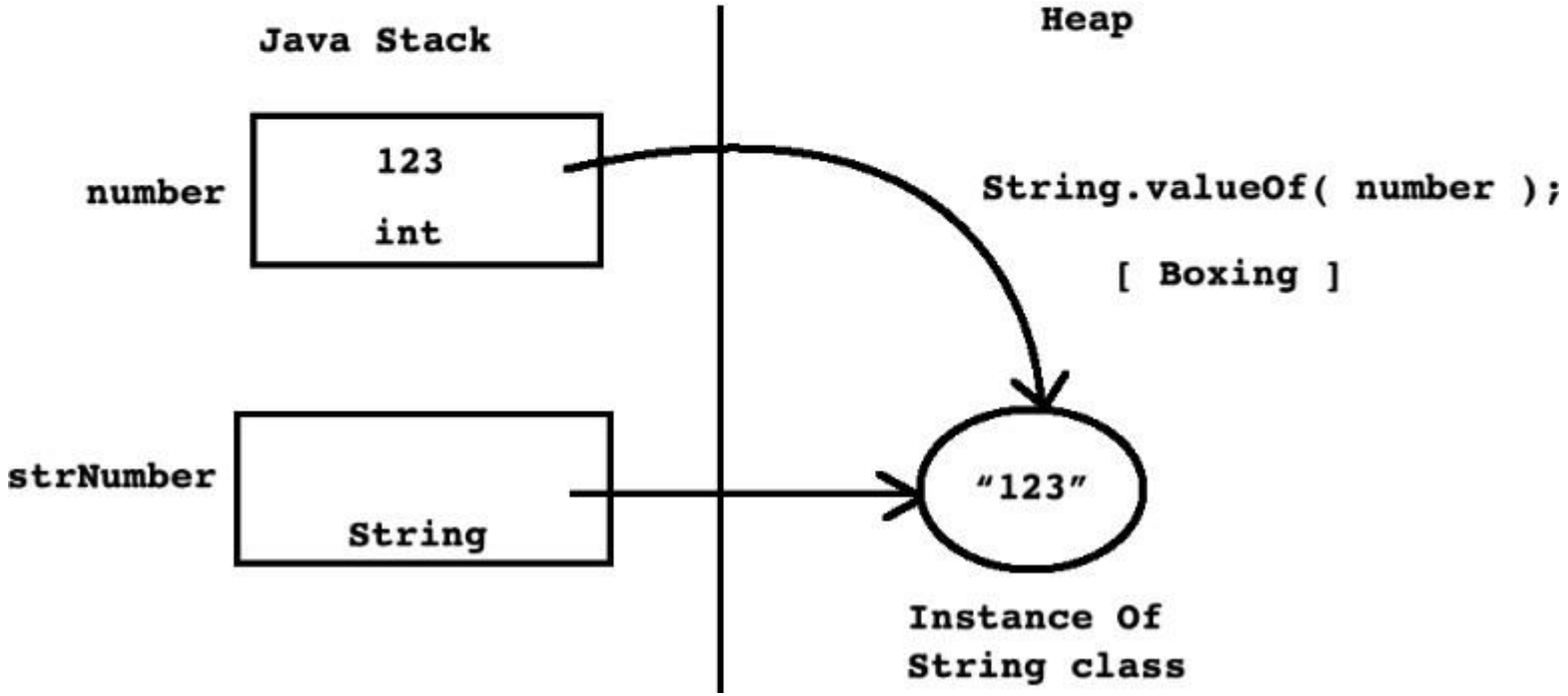
double ---> int

float --> long

double ---> float

Boxing

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



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);

Unboxing

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

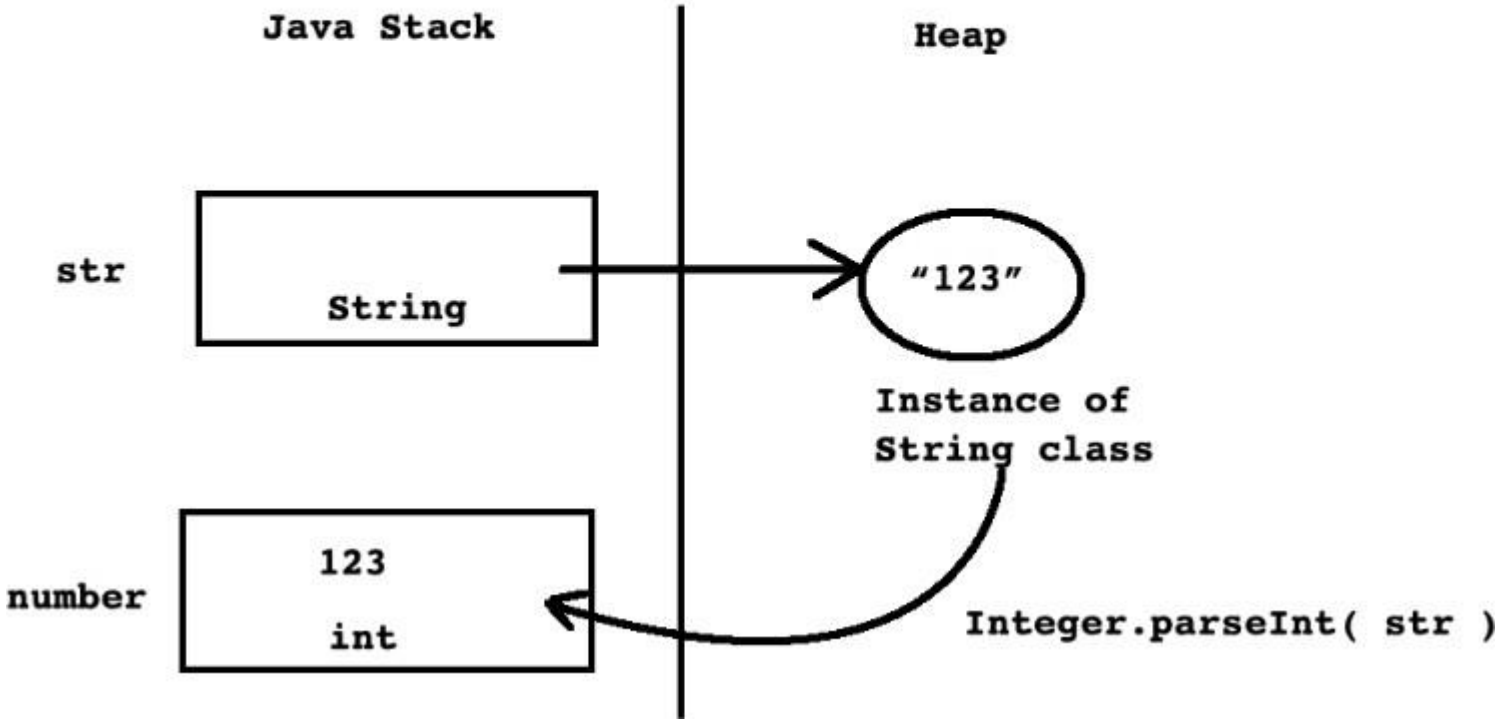
```
public static void main(String[] args) {
    String str = "123";
    int number = Integer.parseInt(str); //UnBoxing
    System.out.println("Number  :   "+number);
}
```

- 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)

Thank you.

Rohan.paramane@sunbeaminfo.com