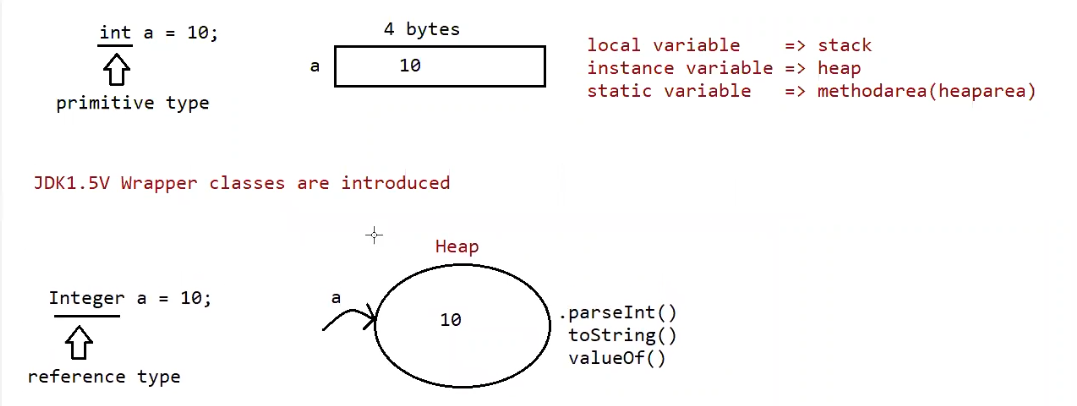
Wrapper classes :



When we create a variable of primitive data type , memory for it is assigned based on whether it is local , instance, static variable.

But when we create a variable of wrapper class the , memory is not allocated as primitive data type , instead a object will be created and data is stored in the object.

The reason for using wrapper classes is it creates object , and with that reference of the object we can call some methods eg: .parseInt() , .toString() , .valueOf() ( to check all the supported methods use command javap java.lang.Integer )

Since the body for methods is already available we can call them as utility methods, and classes as helper classes.

For every class parent happens to be Object , but for wrapper classes the parent happens to be Number class ,And Object is the parent of Number class

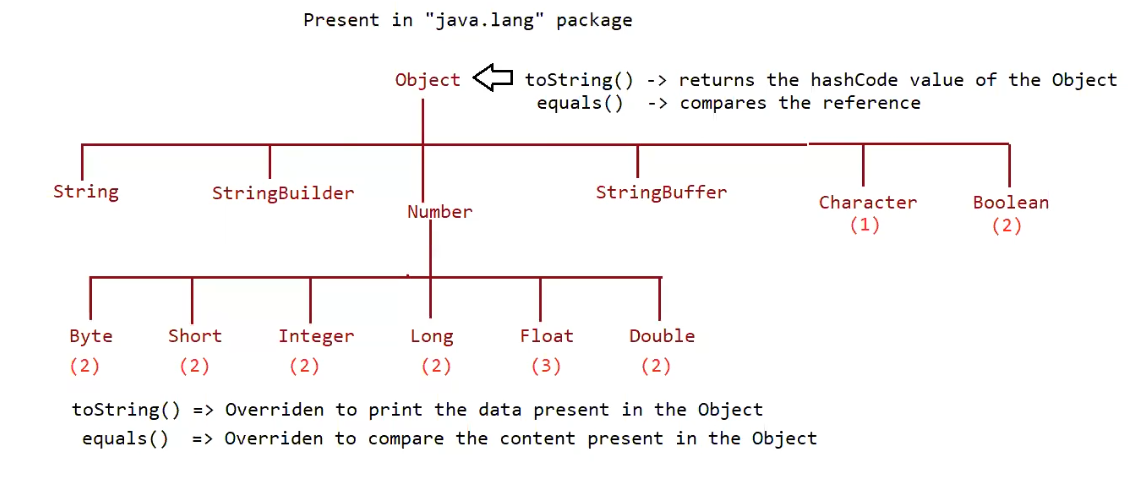
toString() is present in object class. ( to string() returns the hashcode value of the object )

if you are trying to keep the data in the string and if that data is not of integer type, then it would result in the Number Format Exception.

Purpose of wrapper classes :

To wrap primitives into object form ,so that we can handle primitives also just like objects.

To define several utility functions which are required for the primitives



* Almost all the wrapper classes have 2 constructors eg: Byte , Short , Integer , Long , Float , Double.
* But Character and Boolean have only 1 constructor.
* Almost for all the user defined and predefined classes the parent is Object class.
* But for Byte , Short , Integer , Long , Float , Double wrapper classes the parent is Number class.
* Object class contains the toString() , equals() that have above functionality.
* The methods can be overridden by classes and they may have some other functionality .

Example equals() of String class compares the content not the reference.

* The toString() method of wrapper classes are overridden and have the above functionality.

Constructors (check using the command javap java.lang.Integer , and for remaining wrapper classes use the above command just the name after lang.Wrapper\_Class\_Name )

1. One taking primitive type

eg: public java.lang.Integer(int);

Integer a = new Integer(10);

1. One taking String type

Eg: public java.lang.Integer(java.lang.String) throws java.lang.NumberFormatException;

Integer a1 = new Integer(“10”);

Note: If String argument is not properly defined then it would result in RunTimeException called “NumberFormatException”

Eg: Integer I = new Integer(“ten”) // RuntimeException : NumberFormatException .

Wrapper class and its associated constructor

Note : check every class constructors by javap java.lang.Wrapperclass\_Name

Byte => byte and String

Short => short and String

Integer => int and String

Eg: Integer\_Constructors // go through the code.

Long => long and String

Float => float, double and String

Eg: Float\_Constructors // go through the code

// after going through the code check this link.

// <https://docs.oracle.com/javase%2F9%2Fdocs%2Fapi%2F%2F/java/lang/Float.html>

Double => double and String

Character => character

Eg: Character\_Constructors // go through the constructors

Boolean => boolean and String

Eg: Boolean\_Constructors

Note :

Incase of Boolean constructor , boolean value is treated be treated as true w.r.t to case insensitive part of “true” , for all others it would be treated as “false”.

Note:

If we are passing String argument then case is not important and content is not important

If the content is case insensitive String of true , is treated as true in all other cases it is treated as false.

Note:

Incase of wrapper class , toString() is overridden to print the data.

Incase of wrapper class , equals() is overridden to check the content.

Just like String class , Wrapper classes are also treated as “immutable class”.

Eg: Interger\_Equals\_Method

// go through the code

Can we make our user defined class immutable ?

Yes , shown as below .

