VIT-AP UNIVERSITY

Course code : **CSE2005**

Course title : **Object Oriented Programming**

Wrapper Classes

Objectives

This session will give the knowledge about

- Describe the need for wrapper classes
- Define wrapper classes

Wrapper Classes

For all the primitive data types available in Java, there is a corresponding Object representation available which is known as Wrapper Classes

Need for Wrapper Classes

- All Collection classes in Java can store only Objects
- Primitive data types cannot be stored directly in these classes and hence the primitive values needs to be converted to objects
- We have to wrap the primitive data types in a correspondingobject, and give them an object representation

Wrapper Classes

- Definition: The process of converting the primitive data types into objects is called *wrapping*
- To declare an integer ‘i’ holding the value 10, you write int i = 10;
- The object representation of integer ‘i’ holding the value 10 will be:
    - Integer iref = new Integer(i);
- Here, class Integer is the wrapper class wrapping a primitive data type i

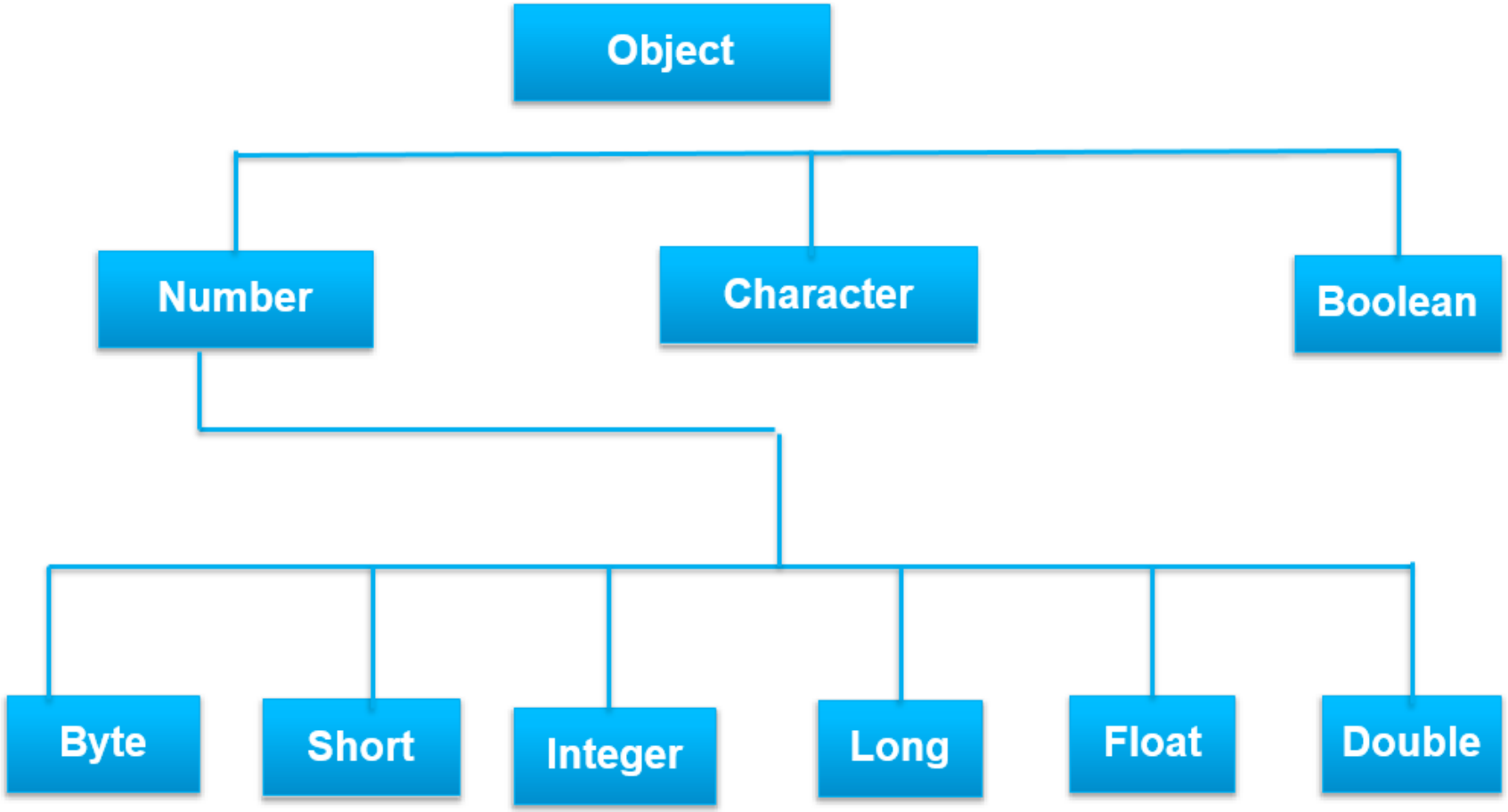
Wrapper Classes

- The Java API has provided a set of classes that make the process of wrapping easier. Such classes are called wrapper classes.
- For all the primitive data types, there are corresponding wrapper classes. Storing primitive types in the form of objects affects the performance in terms of memory and speed.
- Representing an integer via a wrapper takes about 12-16 bytes, compared to 4 in an actual integer. Also, retrieving the value of an integer uses the method Integer.intValue().

Wrapper Classes

- For example, you can take the integer input from the user in the form of a String, and convert it into integer type using the following statements:
    - String str = “100”;
    - int j = Integer.parseInt(str);
- The wrapper classes also have constants like :
- MAX_VALUE, MIN_VALUE, NaN (Not a Number), POSITIVE_INFINITY, and NEGATIVE_INFINITY.

Wrapper Classes



The Integer Class

Class **Integer** is a wrapper for values of type **int. Integer** objects can be constructed with a **int** value, or a string containing a int value

The constructors for Integer are shown here:

- **Integer(int num)**
- **Integer(String str)** throws NumberFormatException

Some methods of the **Integer** class:

- **static int parseInt(String str)** throws NumberFormatException
- **int intValue( )** returns the value of the invoking object as a int value

The Float Class

Class **Float** is a wrapper for values of type **float. Float** objects can be constructed with a **float** value, or a string containing a float value

The constructors for Float are shown here:

- **Float(float num)**
- **Float(String str)** throws NumberFormatException

Some methods of the **Float** class:

- **static float parseFloat(String str)** throws NumberFormatException
- **float floatValue( )** returns the value of the invoking object as a float value

Methods common to All numerical wrapper classes

byteValue()

Returns the value of the invoking object as a byte.

doubleValue()

Returns the value of the invoking object as a double.

floatValue()

Returns the value of the invoking object as a float.

longValue()

Returns the value of the invoking object as a long.

shortValue()

Returns the value of the invoking object as a short.

The Integer Class

public static String toBinaryString(int i)

This method is used to find base 2(with no extra leading 0s (zeros)) for the given int.

public static String toOctalString(int i)

This method is used to find base 8 (with no extra leading 0s (zeros)) for the given int.

public static String toHexString(int i)

This method is used to find base 16(with no extra leading 0s (zeros)) for the given int.

The Integer Class

```
public class Main {
        public static void main(String ar[]){
                int x=46;
                System.out.println(Integer.toBinaryString(x));
                System.out.println(Integer.toOctalString(x));
                System.out.println(Integer.toHexString(x));
        }
}
```

```
101110
56
2e
```

Character Class

- Character class is a wrapper class for character data types.
- The constructor for Character is:

  Character(char c)

- Here, c specifies the character to be wrapped by the Character object
- After a Character object is created, you can retrieve the primitive character value from it using:

  char charValue( )

Character Class

```
/* this is an example to count number of
alphabets
upper case letters
lower case letters
special symbols
digits
white spaces */

public class Main {
        public static void main(String ar[]){
                String input="I am TOM2020!";
                char[] ary=input.toCharArray();
```

Character Class

```
int alpha=0,upper=0,lower=0,symb=0,digi=0,space=0;
for(char c:ary){
        if(Character.isAlphabetic(c))
                alpha++;
        if(Character.isDigit(c))
                digi++;
        if(Character.isUpperCase(c))
                upper++;
        if(Character.isLowerCase(c))
                lower++;
        if(Character.isWhitespace(c))
                space++;
```

Character Class

```
                }
                symb=ary.length-(alpha+digi+space);

                System.out.println("no.of alphabets "+alpha);
                System.out.println("no.of digits "+digi);
                System.out.println("no.of upper case "+upper);
                System.out.println("no.of lower case "+lower);
                System.out.println("no.of spaces "+space);
                System.out.println("no.of symbols "+symb);
        }
}
```

Character Class

no.of alphabets 6
no.of digits 4
no.of upper case 4
no.of lower case 2
no.of spaces 2
no.of symbols 1

The Boolean Class

- The Boolean class is a wrapper class for boolean values
- It has the following constructors:
  - **Boolean(boolean bValue)**
    - Here, bValue can be either true or false
  - **Boolean(String str)**
    - The object created by this constructor will have the value true or false depending upon the string value in str – “true” or “false”
    - The value of str can be in upper case or lower case

Summary

We have discussed about

- Describe the need for wrapper classes
- Define wrapper classes