FAQ's on Wrapper classes

1. What are Wrapper Classes?

In Java, for every primitive type, corresponding class is defined. Such classes are called Wrapper Classes.

Primitive Types	Wrapper Classes
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
char	Character
boolean	Boolean

2. What is the hierarchy of the Wrapper classes?

The following diagram depicts the hierarchy of the Wrapper Classes:

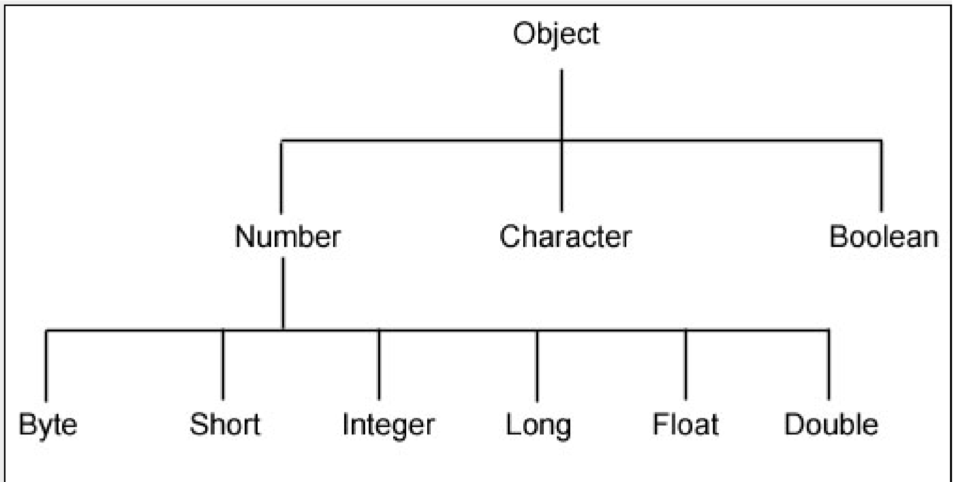

Figure : Wrapper classes Hierarchy

3. Why do we need Wrapper classes?

We need Wrapper Classes:

- In order to represent the primitive type of data into its object type
- For representing every String object into its equivalent primitive type.

4. What is boxing?

It is the process of converting from primitive type into its object type.

Example:
```
   int i=10; //primitive type
   Integer i1 = new Integer(10); //converted into its object type
```

5. What is unboxing?

It is the process of converting from object type to its equivalent primitive type.
```
   Integer i2=new Integer(20); //object type
   int i3=i2.intValue(); //primitive type
```

6. What is auto boxing?

“implicit conversion from primitive type to object type is known as auto boxing”.

(OR)

From java1.5 onwards, the process of converting primitive type to object type is done by JVM internally it is known as “autoboxing”.

7. What is auto Unboxing?

The implicit conversion from a primitive type to an object type is known as auto boxing.

(OR)

Form jdk 1.5 onwards, the process of converting object type into its equivalent primitive type is done by the JVM internally it is known as “auto unboxing”.

Note: Hence we can apply arthematic operations directly on both primitive type and object types based on the type of need JVM will take care of it.

8. Program for Wrapping and unwrapping?

```
Package com.talentsprint.wrapperclasses;
public class WrappingUnwrapping{
public static void main(String args[]){
      byte  grade = 2;
      int  marks = 50;
      float  price = 8.6f;
      double  rate = 50.5;
      Byte g1 = new Byte(grade); // wrapping
      Integer m1 = new Integer(marks);
      Float f1 = new Float(price);
      Double r1 = new Double(rate);
```

```
   // let us print the values from objects
     System.out.println("Values of Wrapper objects (printing
as objects)");
      System.out.println("Byte object g1:  " + g1);
      System.out.println("Integer object m1:  " + m1);
      System.out.println("Float object f1:  " + f1);
      System.out.println("Double object r1:  " + r1);
 // objects to data types (retrieving data types from objects)
      byte bv = g1.byteValue(); // unwrapping
      int iv = m1.intValue();
      float fv = f1.floatValue();
      double dv = r1.doubleValue();
 // let us print the values from data types
      System.out.println("Unwrapped values (printing as data
types)");
      System.out.println("byte value, bv: " + bv);
      System.out.println("int value, iv: " + iv);
      System.out.println("float value, fv: " + fv);
      System.out.println("double value, dv: " + dv);
  }
}
Output:
Values of Wrapper objects (printing as objects)
Byte object g1:  2
Integer object m1:  50
Float object f1:  8.6
Double object r1:  50.5
Unwrapped values (printing as data types)
byte value, bv: 2
int value, iv: 50
float value, fv: 8.6
double value, dv: 50.5
```

9. Compile and run the below code? Mention the reason?

```
package com.talentsprint.wrapperclasses;
public class WrapDemo {
   public static void main(String[] args) {
      int i1=10;
      System.out.println({“\t” +i1);
      Integer i2=new Integer(10);
      System.out.print(“\t” +i2);
      if(i1==i2){
          System.out.print("\t i1 and i2 are same");
      else{
          System.out.print("\t i1 and i2 are not same");
      }
   }
}
```

Output: 10 10 i1 and i2 are same

Reason: "i1 is a local variable of primitive type", so in println() method there is an implementation to print the primitive types as value directly.

"i2 is a local variable of user-defined type ", in all wrapper classes they have overridden the toString() and hashCode() method in such way that it returns content present in it instead of address of i2.

i1==i2 gives true, because it compares the values of primitive types.

10. What is the output of the below code?

```
package com.talentsprint.wrapperclasses;
public class WrapDemo2 {
  public static void main(String args[]){
       String s1="135";
       Integer i1=new Integer(s1);
       Integer i2=new Integer(s1);
       System.out.println(i1.hashCode());
       System.out.println(i2.hashCode());
       System.out.println(i1==i2);
  }
}
Output: 135
        135
        false
```

Reason: Even though i1 and i2 having same values, we are creating them using new operator it will create different address every time.

11. What will be the result on compiling and executing the below the code?

```
package com.talentsprint.wrapperclasses;

public class WrapDemo2 {
  public static void main(String args[]){
        String s1="123abc";
        Integer i1=new Integer(s1);
        System.out.println(i1);
        String s2="123";
        Integer i2=new Integer(s2);
        System.out.println(i2);
   }
}
```

Result:
At line no 6: it is a Runtime exception of NumberFormatException type because it is not a valid integer value.

Line no 10: It prints 123; because it is a valid integer.

12. What will be the result on compiling and executing the below the code?

```
package com.talentsprint.wrapperclasses;
    public class WrapDemo3 {
      public static void main(String args[]){
             String s2="130";
             Byte b1=new Byte(s2);
             System.out.println(b1);
      }
    }
```

Result is: Runtime Exception of type OutOfRangeException ,Bcoz bytre range is upto 127.

13. What will be the result on compiling and executing the below the code?

```
package com.talentsprint.wrapperclasses;
    public class WrapDemo3 {
    public static void main(String args[]){
           String s1="true";
           boolean b1=Boolean.parseBoolean(s1.trim());
           System.out.println(b1);
           String s2="false";
           boolean b2=Boolean.parseBoolean(s2.trim());
           System.out.println(b2);
           String s3="false or true";
           boolean b3=Boolean.parseBoolean(s3.trim());
           System.out.println(b3);
    }
    }

    Result is: true
               false
               false
   (Reason is: boolean treats everything as false other than true)
```

14. Write program to convert String from back to its data types using parseXXX() methods?

```
Parsing Operations example:
Package com.talentsprint.wrapperclasses;
```

```
public class ParsingDemo{
  public static void main(String args[]){
    String price ="100"; // declare some strings
    String rate = "5.8f";
    String tax ="50.2";
    // performing parsing operations on strings
    int x = Integer.parseInt(price);
    float f1 = Float.parseFloat(rate);
    double y = Double.parseDouble(tax);

    System.out.println("\nPrinting data type values after
    parsing");
    System.out.println("int value: " + x);
    System.out.println("float value: " + f1);
    System.out.println("double value: " + y);
    //another style of converting strings into data types,
    very less used
    Integer i1 = new Integer(price);
    Float f2 = new Float(rate);
    Double d1 = new Double(tax);
    // extracting data types from wrapper objects
    int x1 = i1.intValue();
    float f3 = f2.floatValue();
    double d2= d1.doubleValue();

    System.out.println("\nPrinting data type values after
    conversion");
    System.out.println("int value: " + x1);
    System.out.println("float value: " + f3);
    System.out.println("double value: " + d2);
  }
}
Output:
Printing data type values after parsing
int value: 100
float value: 5.8
double value: 50.2

Printing data type values after conversion
int value: 100
float value: 5.8
double value: 50.2
```

15. Is string a wrapper class?

String is a class, but not a wrapper class. Wrapper classes like (Integer) exist for each primitive type. They can be used to convert a primitive data value into an object, and vice versa.

16. Can a Byte object be cast to a double value?

No, an object cannot be cast to a primitive value.

17. What is casting?

There are two types of casting, casting between primitive numeric types and casting between object references. Casting between numeric types is used to convert larger values, such as double values, to smaller values, such as byte values. Casting between object references is used to refer to an object by a compatible class, interface, or array type reference.

18. What is Downcasting?

Downcasting is the casting from a general to a more specific type, i.e. casting down the hierarchy.

19. How many bits are used to represent Unicode, ASCII, UTF-16, and UTF-8 characters?

Unicode requires 16 bits and ASCII require 7 bits although the ASCII character set uses only 7 bits, it is usually represented as 8 bits.

UTF-8 represents characters using 8, 16, and 18 bit patterns.

UTF-16 uses 16-bit and larger bit patterns.

20. What is the range of the char type?

The range of the char type is 0 to 216 - 1 (i.e. 0 to 65535).

21. What is the range of the short type?

The range of the short type is -(215) to 215 - 1. (i.e. -32,768 to 32,767).

22. Name the eight primitive Java types.

The eight primitive types are byte, char, short, int, long, float, double, and boolean.

23. What is numeric promotion?

Numeric promotion is the conversion of a smaller numeric type to a larger numeric type, so that integer and floating-point operations may take place. In numerical promotion, byte, char, and short values are converted to int values. The int values are also converted to long values, if necessary. The long and float values are converted to double values, as required.

24. To what value is a variable of the boolean type automatically initialized?

The default value of the boolean type is false.

25. What are wrapper classes?

Java provides specialized classes corresponding to each of the primitive data types. These are called wrapper classes.

They are example: Integer, Character, Double etc.

26. Why do we need wrapper classes?

It is sometimes easier to deal with primitives as objects. Moreover most of the collection classes store objects and not primitive data types. And also the wrapper classes provide many utility methods also.

Because of these reasons we need wrapper classes. And since we create instances of these classes we can store them in any of the collection classes and pass them around as a collection. Also we can pass them around as method parameters where a method expects an object.

27. Is String a primitive data type in Java?

No. String is not a primitive data type in Java, even though it is one of the most extensively used objects. Strings in Java are instances of String class defined in `java.lang` package.