Wrapper Classes

Exercises and solutions

1. What are wrapper classes in Java? Name the wrapper classes for the following primitive types: byte, int, long, and char.

**Answer:**

Java provides eight classes in java.lang package to represent the values of each of the eight primitive types as objects. These classes are called wrapper classes as they wrap a primitive value in an object.

The name of the wrapper classes for the primitive types: byte, int, long, and char are Byte, Integer, Long, and Character, respectively.

1. Using the wrapper class Integer, print the maximum and minimum values of the int data type.

**Solution:**

System.out.println("Integer max value = " + Integer.MAX\_VALUE);

System.out.println("Integer min value = " + Integer.MIN\_VALUE);

1. What is the name of the superclass of the numeric wrapper classes?

**Answer:**

java.lang.Number

1. Suppose you have a string "1969". Complete the following snippet of code to store the integer value in the string into an int variable and an Integer object.  
     
   String str = "1969";  
   int value = /\* Your code goes here \*/;  
   Integer object = /\* Your code goes here \*/;

**Solution:**

String str = "1969";  
int value = Integer.parseInt(str);  
Integer object = Integer.valueOf(str);

1. You have a string "7B1", which contains an integer in hexadecimal format. Use the Integer class to parse and store its value in an int variable.

**Solution**

int value = Integer.parseInt("7B1", 16);

1. Will the following snippet of code compile? If it will, describe the rules/reasons.  
     
   Integer x = 19;

**Answer:**

Yes. This snippet of code will compile fine. The value 19 is of the int data type, which is automatically wrapped in an Integer object and assigned to variable x, using autoboxing.

1. You have an integer value of 1969 and you want to print its value in hexadecimal format. Complete the following snippet of code that achieves this:  
     
   int x = 1969;  
   String str = Integer./\* your code goes here \*/;  
   System.out.println("1969 in hex is " + str);

**Answer:**

String str = Integer.toHexString(x);

1. Why does the following statement not compile  
     
   Double x = 1969;  
     
   and the following statement does?  
     
   double y = 1969;  
     
   Make sure you understand the reasons behind these statements being invalid and valid. Describe how the following statement compiles.   
     
   Number x = 1969;

**Answer:**

Double x = 1969;

By autoboxing, the primitive int value 1969 becomes an Integer object. The compilation fails because the variable x is of type Double and the right-hand-side is an Integer object. Integer to Double assignment is not allowed.

double y = 1969;

In this case, the primitive int value 1969 is widened using automatic widening conversion and is assigned to the double variable y.

Number x = 1969;

By autoboxing, the primitive int value 1969 becomes an Integer object. All numeric wrapper classes such as Integer are inherited from the Number class. Using the subclassing rules of assignment, an Integer is assignment compatible to a Number. This is the reason why this statement compiles fine.

1. What will be the output of the following snippet of code? Explain your answer.  
     
   Number x = 1969;  
   System.out.println(x.getClass().getSimpleName());

**Answer:**

Integer

Variable x is of type Number, which is an abstract class. Abstract class cannot be instantiated, but can be assigned any object reference of its subclasses. The runtime type of x is java.lang.Integer because of autoboxing of the int value 1969. This is the reason the output is Integer.

1. What will be the output when the following snippet of code is run?

Double x = 128.5;  
System.out.println(x.intValue());  
System.out.println(x.byteValue());

**Answer:**

128

-128