CORE JAVA

MANUAL V8.3

MODULE CODE:

ANUDIP FOUNDATION

ICONS AND THEIR MEANING
HINTS:
Get ready for helpful insites on difficult topics and questions.
STUDENTS:
This icon symbolize important instrcutions and guides for the students.
TEACHERS/TRAINERS:
This icon symbolize important instrcutions and guides for the trainers.

Module 3: String Handling, Regular Expression and Wrapper class

Chapter 4

Objective: After completing this lesson you will be able to : * Gain an understanding of Java wrapper classes * Learn about the usages of Double, Integer and Float wrapper classes * Learn the predefined functions of these wrapper classes	**Materials Required:** 1. Computer 2. Internet access
Theory Duration: 90 minutes	**Practical Duration:** 30 minutes
Total Duration: 120 minutes	

Chapter 4

4.1 Wrapper Class

In Java, a wrapper class is a class that has objects containing or wrapping primitive data types. An object created within a wrapper class has a field capable of storing primitive data types. A wrapper class object can be used to wrap a primitive value. A wrapper class can be utilized to convert any type of data into an object.

The importance of wrapper classes

1. Wrapper classes convert primitive data types into objects. Objects are required for modifying arguments passed through methods.
2. Wrapper classes are also necessary as java.util packages can only operate with objects.
3. Multithreading synchronization requires objects converted by wrapper classes.
4. Collection framework data structures store only objects, but cannot store primitive data types.

Primitive data types and their wrapper classes

Data type	**Wrapper class**
boolean	Boolean
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double

Example of a wrapper class program –

```
public class WrapperExample{
        public static void main(String args[]){

        int c=18;


        Integer j=Integer.valueOf(c);

        Integer k=c;

        System.out.println(c+' '+j+' '+k);

        }}
```

Output: 18 18 18

4.2 Usage of Wrapper Class-Integer, Float, Double

*** Integer**

The Java integer class (java.lang.Integer) is a wrapper class for the primitive data type 'int'. It contains methods for converting 'int' values into strings and strings into 'int' values. An Integer wrapper class object can hold only one int value.

Integer wrapper class constructors:

* **Integer(int c)**: Used for creating an integer object with the given value

Syntax – public Integer(int c)

* **Integer(String s)**: Used for creating an integer object with given string-represented int value

Syntax – public Integer(String s)

Example of the integer wrapper class in use –

```
public class IntegerExample {
  public static void main(String[] args) {
→ Using int for creating an integer object –
    Integer intObj1 = new Integer(15);
 → Using String for creating an Integer object –
    Integer intObj2 = new Integer('15');
    System.out.println(intObj1);
    System.out.println(intObj2);
  }
}
```

Output:

```
15
15
```

*** Float**

The Java float class (java.lang.Float) is a wrapper class for the primitive data type 'float'. It contains methods for converting 'float' values into strings and strings into 'float' values. A float wrapper class object can hold only one float value.

Float wrapper class constructors:

*** Float(float c):** Used for creating a float object with given value

Syntax – public Float(float c)

*** Float(String s):** Used for creating a float object with given string-represented parsed float value

Syntax – public Float(String s)

Example of the float wrapper class in use –

```
import java.util.*;
public class JavaFloatExample {
        public static void main(String[] args) {
        float <variable name> = <default value>;
                float f = 12.2f;
                System.out.println('Float variable f value is:' + f);
        }
}
}
```

Output:

```
Float variable f value is: 12.2
```

*** Double**

The Java double class (java.lang.Double) is a wrapper class for the primitive data type 'double'. It contains methods for converting 'double' values into strings and strings into 'double' values. A double wrapper class can hold a solitary 'double' value.

Double wrapper class constructors:

*** Double(double c):** Used for creating a double object with the given value

Syntax – public Double(double c)

*** Float(String s):** Used for creating a double object with given string-represented parsed double value

Syntax – public Double(String s)

Example of the double wrapper class in use –

```
public class JavaDoubleExample {
            public static void main(String[] args) {
   --| Creating a Double object using double primitive type –
    double c = 20.20;
    Double cObj1 = new Double(c);
    System.out.println(cObj1);

    --| Creating Double object using String –
    Double cObj3 = new Double('45.48');
    System.out.println(cObj3);

  }
}
```

Output:

```
20.2
45.48
```

Predefined Functions of Each Wrapper Class

* **Integer predefined functions**

1. parseInt(s) – returns a decimal integer with a value equal to string s

2. toString(i) – returns new string object resembling integer i
3. byteValue() – returns integer value as a byte
4. doubleValue() – returns integer value as a double

5. floatValue() - returns integer value as a float

6. ntValue() - returns integer value as an int

7. shortValue() - returns integer value as short

8. longValue() - returns integer value as long

9. int compareTo(int i) - Compares object numerical value to integer i’ s value. It returns 0 for equal values, positive if invoking object has higher value, and negative if invoking object value is lower.

10.static int compare(int num1, int num2) - Compares num1 and num2 values. It returns 0 for equal values, negative value if num 1 < num 2, and positive value if num 1 ı num 2.

11. boolean equals(Object intObj) - Returns a true output if integer object and intObj are equal. Returns false if values are not equal.

*** Float predefined functions**

1. doubleValue() - Returns double of a float value

2. equals(Object) - Compares an object with another

3. floatToIntBits(float) - Returns single-float value bit representation

4. floatValue() - Returns the Float object’ s float value.

5. hashCode() - Returns a Float hashcode

6. intBitsToFloat(int) - Returns a bit representation’ s matching single-float value

7. intValue() - Returns a Float integer value

8. isInfinite(float) - Returns a true output if a number’ s value is considered infinite

9. isInfinite() - Returns a true output if a Float value is considered infinite

10. isNaN(float) - Returns true output if a number is equivalent to the Not-a-Number (NaN) value.

11. isNaN() - Returns true output if a Float value is Not-a-Number (NaN).

12. longValue() - Returns the Float’s long value

13. toString(float) - Returns a float value’s string representation

14. toString() - Returns a Float object’s string representation

15. valueOf(String) - Returns the string-specified floating point value

*** Double predefined functions**

1.doubleToLongBits(double) - returns a double-float value's bit representation

2. doubleValue() - returns a Double's double value

3. equals(Object) - Compares an object against another particular object.

4. floatValue() - Returns a Double's float value.

5. hashCode() - Returns a Double's hashcode

6. intValue() - Returns a Double's integer value

7. isInfinite(double) - Returns a true output in case a number has infinite value

8. isInfinite() - Returns a true output if Double has infinite value

9. isNaN(double) - Returns a true output if a number is Not-a-Number (NaN) value.

10. isNaN() - Returns a true output if a Double value is Not-a-Number (NaN) value.

11. longBitsToDouble(long) - Returns a matching double-float value for a bit representation.

12. longValue() - Returns the Double' s long value

13. toString() - Returns a Double object' s string representation

14. valueOf(String) - Returns new value for Double initialized to the string-specified value representation

Practical (30 minutes)

See the example programme for Java Wrapper Class below. Write the same programme for the class WrapperDemo1 with three integers a, b and c, where a, b and c are equal. Show the resulting output.

```
public class WrapperExample{

        public static void main(String args[]){

        int c=18;

        Integer j=Integer.valueOf(c);

        Integer k=c;
        System.out.println(c+‘ ‘+j+‘ ‘+k);

        }}
```

Instructions: The progress of students will be assessed with the exercises mentioned below.

MCQ (10 minutes)

1. Java wrapper class objects wrap __________ data types.

a) primitive

b) non-primitive

c) integer

d) None of the mentioned

2. Wrapper classes convert primitive data types into ________.

a) arrays

b) objects

c) arguments

d) None of the mentioned

3. What type of synchronization requires objects converted by wrapper classes.

a) multithreading

b) single threading

c) core threading

d) None of the mentioned

4. java.lang.Integer class can convert int values to __________.

a) Arrays

b) threads

c) strings

d) None of the mentioned

5. An Integer wrapper class _______ is capable of holding only one int value.

a) object

b) program

c) structure

d) None of the mentioned

6. An int c constructor can be used to create _____________ object.

a) a clone

b) a decimal

c) an integer

d) None of the mentioned

7. java.lang.Float can be used to convert strings into _______ values.

a) int

b) float

c) object

d) None of the mentioned

8. Float(String s) creates a _______ ______.

a) float string

b) float thread

c) float object

d) None of the mentioned

9. A double class in Java converts ________ values into ________.

a) single, doubles

b) double, singles

c) double, strings

d) None of the mentioned

10. An integer floatValue() returns an integer ____ as a float.

a) Value

b) type

c) object

d) None of the mentioned