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 2: Object Oriented Programming and Package

Chapter 7

Objective: After completing this lesson you will be able to : * Understand the concepts of Java boxing and unboxing * Gain knowledge about Java garbage collection and finalize	**Materials Required:** 1. Computer 2. Internet access
Theory Duration: 90 minutes	**Practical Duration:** 20 minutes
Total Duration: 120 minutes	

Chapter 7

7.1 Boxing and un-boxing

* Boxing

Boxing or autoboxing in Java is the process by which a primitive data type is converted into an object of the matching wrapper class. It converts boolean, byte, char, short, int, long, float and double data types into their respective wrapper class objects.

Java autoboxing is applicable when –

1. Primitive value is passed as a method parameter for a method that requires a matching wrapper class object. For instance, a method containing a character argument can be called by passing char. The compiler converts a char into a character object.
2. Primitive value is allocated a variable of its matching wrapper class.

Example of autoboxing in Java –

```
class BoxingExample1{
        public static void main(String args[]){
        int a=40;
        Integer b2=new Integer(a);
        Integer b3=4;
        System.out.println(b2+" "+b3);
```

```
    }

    }
```

Output:

```
40 4
```

*** Unboxing**

Unboxing or auto-unboxing is the opposite of autoboxing, as it converts a wrapper type object into its matching primitive data type. The compiler automatically extracts an object value from a wrapper type class.

Java unboxing is applicable when –

1. Wrapper class is passed as a method parameter for a method that requires a matching primitive data type.
2. Wrapper class is allocated a variable of its matching primitive data type.

Example of unboxing in Java –

```
class UnboxingExample1{
public static void main(String args[]){
Integer i=new Integer(30);
int a=i;
System.out.println(a);
}
}
```

Output:

30

7.2 Garbage Collection and Finalize

i) Garbage Collection

The term garbage refers to unreferenced objects in Java.

Garbage collection is the procedure of automatically recovering the unused runtime memory. The process gets rid of all unwanted objects. Garbage collection occurs during a program life cycle, effectively minimizing the necessity of memory de-allocation. This process helps to reduce memory leak possibilities. De-allocating memory also reduces the chances of programs crashing.

The garbage collector program in Java performs automatic memory management. Java itself handles object de-allocation, reducing the burden on programmers. The collector clears the heap memory by removing unreferenced objects.

Java garbage collection example –

```
public class TestGarbage1{

    public void finalize(){System.out.println('garbage collected object');}
    public static void main(String args[]){
    TestGarbage1 s1=new TestGarbage1();
    TestGarbage1 s2=new TestGarbage1();
```

```
s1=null;
s2=null;
System.gc( );
 }
}
```

Output:

garbage collected object

garbage collected object

ii) Finalize

The finalize method in Java is called by the garbage collector right before destroying an object suitable for collection. It helps in shutting down resources or performing 'clean up' before the garbage collection process. Finalize makes sure that there are no more references to an object. All objects created in Java have a finalize ( ) method. The method is overridden by a subclass for freeing up system resources and retaining the necessary operations.

Example of finalize in Java –

```
class Hello {
   public static void main(String[] args)
   {
      String s = new String('RR');


      s = null;
      System.gc( );
      System.out.println('Main has Completed');
```

```
    }
     public void finalize( )
    {
        System.out.println('finalize method overridden');
    }

}
```

Output: Main has Completed

*** Difference between final, finally and finalize –**

Final	Finally	Finalize
Final is a Java keyword and access modifier.	Finally is a Java block	Finalize is a Java method
It is applicable for methods, classes and variables	It is related to try-catch block in Java	This method can be applied to objects
Final method executes when called.	Finally executes after try-catch block execution.	Finalize method executes prior to an object being destroyed.
Main purpose of final keyword use is to assign limitations on method, variable and class.	It helps in introducing code regardless of if an exception is handled.	Finalize method performs clean up before garbage collection.

Practical (20 minutes)

See the example programme for Java garbage collection below. Write the same programme for a class named GarbageTesting to produce the output "objected collected by garbage collector".

```
public class TestGarbage1{
```

```
    public void finalize( ){System.out.println(‘garbage collected object‘ );}

public static void main(String args[ ]){

TestGarbage1 s1=new TestGarbage1( );

TestGarbage1 s2=new TestGarbage1( );

s1=null;

s2=null;

System.gc( );

}

}
```

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

MCQ

1. _____ in Java is the process by which a primitive data type is converted into an object.

a) ultra-boxing

b) autoboxing

c) unboxing

d) None of the mentioned

2. Autoboxing converts data types into objects with matching _____ classes.

a) wrapper

b) array

c) subject

d) None of the mentioned

3. Which Java component helps to perform autoboxing and unboxing?

a) interpreter

b) compiler

c) packer

d) None of the mentioned

4. What process can be used to convert a boolean data type into a boolean object?

a) autoboxing

b) inboxing

c) unboxing

d) None of the mentioned

5. Autoboxing is applicable when a primitive value is assigned a matching wrapper class _____.

a) variable

b) variant

c) object

d) None of the mentioned

6. Which process is used for converting an object to its corresponding data type?

a) outboxing

b) inboxing

c) unboxing

d) autoboxing

7. A Java compiler automatically extracts an object _____ from a wrapper type class

a) property

b) value

c) inheritance

d) None of the mentioned

8. What kind of objects do garbage refer to in Java?

a) unqualified

b) unreferenced

c) incongruent

d) None of the mentioned

9. Garbage collection helps in recovering unused _____ memory.

a) interval time

b) compile time

c) runtime

d) None of the mentioned

10. The Java finalize method is called by the garbage collector _____ destroying an object.

a) Before

b) after

c) during

d) None of the mentioned