Unit 4: More on class and object

4

Unit Structure

4.1 Learning Objectives

4.2 Visibility control

4.3 public access

4.4 friendly access

4.5 protected access

4.6 private access

4.7 Rules of thumb

4.8 Object as parameters

4.9 Returning Objects

4.10 Recursion

4.11 Nested and inner class

4.12 String class

4.13 StringBuffer class

4.14 Command line argument

4.15 Generic in Java

4.16 Let us sum up

4.17 Check your Progress

4.18 Check your Progress: Possible Answers

4.19 Further Reading

4.20 Assignments

4.1 LEARNING OBJECTIVE

After studying this unit student should be able to:

- Understand and use of access modifiers.
- Use of recursion in java program.
- Input data using command line argument.
- To manipulate the string using String and StringBuffer class
- Understand various types of inner class and its usage.

4.2 VISIBILITY CONTROL

Visibility control means controlling the access of java class, data members and methods of class, constructor, and variables. The visibility control can be implemented with the help of access modifier. The access modifier restrict the access of class, constructor, data members and methods of the class and variables in its scope. There are four access modifiers in java:

1. Default – no keyword specified
2. Private- using private keyword.
3. Protected- using protected keyword
4. Public- using public keyword.

Access modifier / Scope	Default	private	protected	public
In same class	Yes	Yes	Yes	Yes
In child class of same package	Yes	No	Yes	Yes
In other class of same package	Yes	No	Yes	Yes
In child class of other package	No	No	Yes	Yes
In other class of other packages	No	No	No	Yes

Table-8 Scope of aceess modifiers

4.3 PUBLIC ACCESS

The variable, class, and methods must be declared public using public access modifier. for this it uses the keyword public. This access specifier has the widest scope. The public class, methods or variables can be access from everywhere. There is no restriction for public data. The public class, method and variable can be access within class, sub class, class outside the package and class within the package.

```
class A
{
        public int a;
        public A() { a = 0; }
        public A(int x) { a = x; }
        public void printA()
        {
        System.out.println(“ a = “ + a);
        }
}
public class ExDefault          // if a class containing main method is public, we have to
                                //create that class name.java file for that program.
                                // Ex:  ExDefault.java for this program
{
        public static void main(String args[])
        {
        A a1 = new A(9);
        A1.printA();
        }
}
```

4.4 FRIENDLY ACCESS

It is also called Default access. When no access modifier used to declare any class, method or data member, it has friendly access. They can only be accessed within all classes of the same package i.e. the package in which the class is created.

For Example,

```
class A
{
        int a;
        A() { a = 0; }
        A(int x) { a = x; }
        void printA()
        {
        System.out.println(" a = " + a);
        }
        }
        class ExDefault
        {
        public static void main(String args[])
        {
        A a1 = new A(9);
        A1.printA();   //printA can be access other class in same package
        //similarly we can access data member a also.
        }
}
```

4.5 PROTECTED ACCESS

It uses protected keyword to assign protected access modifier. The methods and member variables of a class can be declared protected. It means they can be access within class, within package, and only subclass of the package and subclass of the outside package.

For Example,

```
class A
{
        protected int a;
        A() { a = 0; }
        A(int x) { a = x; }
        protected void printA()
        {
        System.out.println(" a = " + a);
        }
}
class B extends A
{
        B(){ super(); }
        B(int x) { super(x); }
        void printB()
        {
        printA(); //can access in subclass of A, as it is protected
        }
}
class ExDefault
{
public static void main(String args[])
{
A a1 = new A(9);
A1.printA();   //printA can be access other class in same package as it is protected
a1.a=10;   //we can access data member a here because it is protected.
}
}
```

4.6 PRIVATE ACCESS

The private keyword is used to declare private access modifier. The methods and member variables of class declared as private can be access within class only.

For Example,

```
class A
{
        private int a;
        A() { a = 0; }
        A(int x) { a = x; }
        void printA()
        {
                System.out.println(“ a = “ + a);
        }
}
class ExDefault
{
        public static void main(String args[])
        {
        A a1 = new A(9);
        A1.printA();   //printA can be access other class in same package
        a1.a=10;   //we can not access data member a here because it is private.
        }
}
```

4.7 RULE OF THUMB

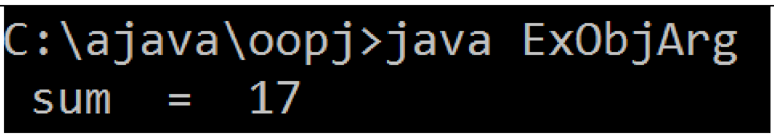
- All important member variables of class should be declared private.
- The final variable should declare public.
- The methods can be declared private only when you want not others to access it.
- Private and protected access modifier should not be used for top level classes. They can be used for inner class declared in nested classes.

4.8 OBJECT AS PARAMETERS

In java, class can have member functions and constructors defined in it. We can pass various arguments to this function. The arguments can be various primitive data types, array or objects. We can pass object of any class as an argument to the function.

For example,

```
class A{
        int a;
        A() { a = 0; }
        A(int x) { a = x; }
        void printA()
        {
        System.out.println(“ a  =  “  +  a);
        }
}
public class ExObjArg
{
        public static void main(String args[])
        {
        A a1 = new A(9);
        A a2 = new A(8);
        add(a1 , a2);
        }
        static void add(A a1, A a2) //function with objects as arguments
        {
        int sum = a1.a + a2.a;
        System.out.println(“ sum  =  “  +  sum);
        }
}
```



```
C:\ajava\oopj>java ExObjArg
 sum  =  17
```

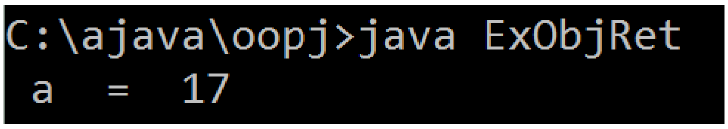
Figure-41 Output of program

4.9 RETURNING OBJECT

A member function of a class can also return an object of any class. The return type of that function must be class type.

For example,

```
class A
{
        int a;
        A() { a = 0; }
        A(int x) { a = x; }
        void printA()
        {
        System.out.println(“ a  =  “  +  a);
        }
}
public class ExObjArg
{
  public static void main(String args[])
  {
  A a1 = new A(9);
  A a2 = new A(8);
  A a3 = add(a1 , a2);
  a3.printA();
  }
  static A add(A a1, A a2)   //function with objects as arguments and returns object
  {
  A a3 = new A();
  a3.a = a1.a + a2.a;
  return a3;
  }
}
```



```
C:\ajava\oopj>java ExObjRet
 a  =  17
```

Figure-42 Output of program

4.10 RECURSION

A function can call itself within its definition. Such function is called recursive function. Programming approach which solves problems using recursive function is called recursion.

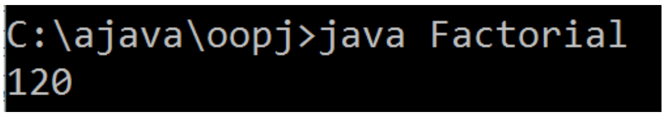
For example,

```
long  factorial( int n)
{
        long fact = 1;
        if( n == 1 || n == 0)
         return fact;
        else
        fact = n * factorial(n-1);
        return fact;
}
```

Example

```
public class Factorial
{
        public static void main(String args[])
        {
        long f = factorial(5);
        System.out.println(f);
        }
        static long  factorial( int n)
        {
        long fact = 1;
        if( n == 1 || n == 0)
         return fact;
        else
        fact = n * factorial(n-1);

        return fact;
        }
}
```



```
C:\ajava\oopj>java Factorial
120
```

Figure-43 Output of program

4.11 NESTED AND INNER CLASS

In java, class can also be created within a class. This is called nested class. Here the class which holds class definition inside it is called outer class and the class inside the outer class is called inner class.

Nested class can be categorized into two. Non-static nested class and static nested class. In nonstatic nested class, the inner class is not static. In static nested class the inner class is declared as static.

4.11.1 NON STATIC NESTED CLASS

Non static nested class are also categorized into three types

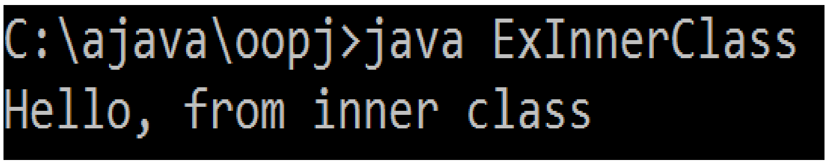
1) Inner class
2) Method local inner class
3) Anonymous inner class

➢ **Inner class**

It is simple to create an inner class. We have to create a class definition inside the class. The inner class can be declared private or public. If it is declared private, we cannot access it outside the outer class. We have to use inner class within the outer class only. The public inner class can be access outside the outer class and the other class also.

For example (private inner class),

```
class OuterClass {
   int n;
   private class InnerClass {
      public void sayHello() {
         System.out.println("Hello, from inner class");
      }
   }
     void useInner() {
      InnerClass x = new InnerClass();
      x.sayHello();
   }
}
   public class ExInnerClass {
   public static void main(String args[]) {
      OuterClass a = new OuterClass();
      a.useInner();
   }
}
```



```
C:\ajava\oopj>java ExInnerClass
Hello, from inner class
```

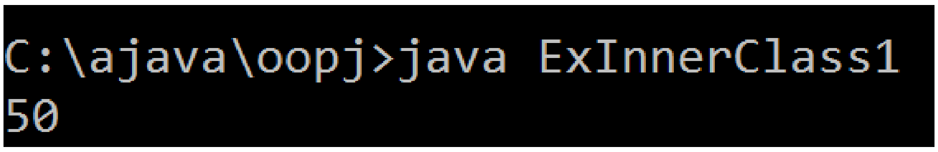
Figure-44 Output of program

For example (public inner class),

In this example the inner class is used to get private member variable of the outer class.

```
class OuterClass {
   private int n = 50;
   public class InnerClass {
      public int getN() {
```

```
      return n;
    }
  }
}
public class ExInnerClass1 {
  public static void main(String args[]) {
    OuterClass x = new OuterClass();
    OuterClass.InnerClass y = x.new InnerClass();
    System.out.println(y.getN());
  }
}
```



```
C:\ajava\oopj>java ExInnerClass1
50
```

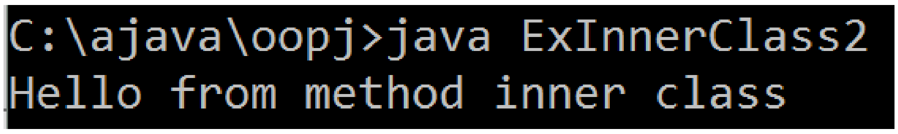
Figure-45 Output of program

➢ **Method local inner class**

In this type of inner class, we can create a class within a function. This class will be the local to the method. This class can be used only inside the method.

For example,

```
class OuterClass {
  void innerFun() {
    int n = 50;
    class InnerClass {              //class inside the method innerFun()
      public void sayHello() {
       System.out.println("Hello from method inner class ");        }
    }
    InnerClass x = new InnerClass();
    x.sayHello();
  }
}
public class ExInnerClass2
{
  public static void main(String args[]) {
    OuterClass y = new OuterClass();
    y.innerFun();
  }
}
```



```
C:\ajava\oopj>java ExInnerClass2
Hello from method inner class
```

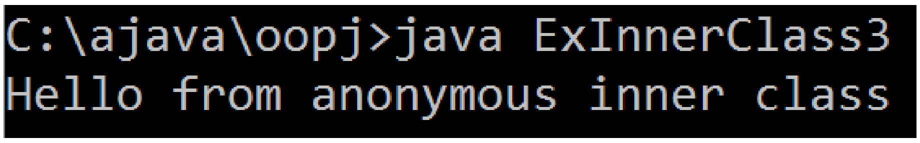
Figure-46 Output of program

➢ **Anonymous inner class**

This type of inner class is use to declare without class name. They are declared and instantiate simultaneously. They are mainly used to override the abstract methods of class or interface.

Example,

```
abstract class InnerClass {
  public abstract void sayHello();
}
public class ExInnerClass3 {
  public static void main(String args[]) {
    InnerClass x = new InnerClass() {
      public void sayHello() {
        System.out.println("Hello from anonymous inner class");
      }
    };
    x.sayHello();
  }
}
```



```
C:\ajava\oopj>java ExInnerClass3
Hello from anonymous inner class
```

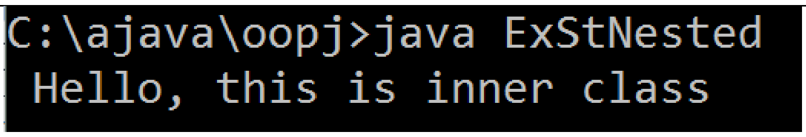
Figure-47 Output of program

4.11.2 STATIC NESTED CLASS

In static nested class, the inner class is declared; hence it is a static member of the outer class. To access this class, the object of outer class is not required. This static inner class can not access the member variables and methods of outer class.

For example,

```
class OuterClass
{
        static class InnerClass
        {
        void sayHello()
        {
        System.out.println(" Hello, this is inner class");
        }
        }
}
public class ExStNested
{
        public static void main(String args[])
        {
        OuterClass.InnerClass x=new OuterClass.InnerClass();
        x.sayHello();
        }
}
```



```
C:\ajava\oopj>java ExStNested
 Hello, this is inner class
```

Figure-48 Output of program

4.12 STRING CLASS

Strings are the sequence of characters. We can store name, address etc. as string. In java string is treated as an object of String class which is available in java.lang package. String class has various methods using which we can create and manipulate the strings.

To create a string in java program following syntax is used.

```
String s="Hello";
```

Here "Hello" is a string literal. For each string literal java compiler creates a String object. We can create a String object using new keyword and constructor. In java strings are non-mutable (non modifiable).

```
String s=new String("Hello");
```

We can also create a String from an array of characters.

```
char[] s1 = { 'h', 'e', 'l', 'l', 'o', '.' };
                    String s = new String(s1);
```

Functions of String class

The following is the list of methods of String class

1) **char charAt(int index)** : this function returns the character at the index position
2) **int compareTo(String str)** : it compares a String with the other String we pass as an argument lexicographically and return difference of those two strings.
3) **int compareToIgnoreCase(String str)** : it compares strings , ignoring case.
4) **String concat(String str)** : it concatenate a string with the specified string passed as an argument.
5) **boolean contentEquals(StringBuffer sb)** : returns true if content of String and StringBuffer is same.
6) **static String copyValueOf(char[] data)** : returns a String having sequence of characters stored in an array.
7) **static String copyValueOf(char[] data, int offset, int count)**: returns a String having count number of characters stored in an array starting from offset.
8) **boolean endsWith(String suffix)** : returns true if String ends with specified String.

9) **boolean equals(String str)** :returns true is both String objects have same content.

10) **boolean equalsIgnoreCase(String anotherString)** : returns true if both String are equal ignoring case. Ex: Hello and hello are equal for this function.

11) **byte getBytes()** : returns a byte array containing String characters.

12) **int hashCode()** : returns a hashcode of the String

13) **int indexOf(int ch)** : returns position of character ch(first occurrence) in the String.

14) **int indexOf(int ch, int fromIndex)** : returns position of character ch in the String after fromIndex.

15) **int indexOf(String str)** : returns position of String str(first occurrence) in the String.

16) **int indexOf(String str, int fromIndex)** : returns position of String str in the String after fromIndex.

17) **int lastIndexOf(int ch)** :returns the position of last occurrence of ch in String.

18) **int lastIndexOf(int ch, int fromIndex)** : returns the position of last occurrence of ch in String. It searches in backward starting from the fromIndex.

19) **int lastIndexOf(String str)** : returns the position of last occurrence of str in String.

20) **int lastIndexOf(String str, int fromIndex)** : returns the position of last occurrence of str in String. It searches in backward starting from the fromIndex.

21) **int length()** :returns the length of the String.

22) **String replace(char oldChar, char newChar)** : returns a new String in which oldChar is replace with newChar in specified String.

23) **boolean startsWith(String prefix)** :returns true if String start with strings specified by prefix.

24) **String substring(int beginIndex)** :return a new String that is a substring start with beginIndex till the end.

25) **String substring(int beginIndex, int endIndex)** : return a new String that is a substring start with beginIndex till the endIndex.

26) **char[] toCharArray()** :converts a string into character array.

27) **String toLowerCase()** : returns a new String which is lower case conversion of specified String.

28) **String toUpperCase()** : returns a new String which is upper case conversion of specified String.

29) **String trim()** :returns a copy of String after removing starting and ending spaces.

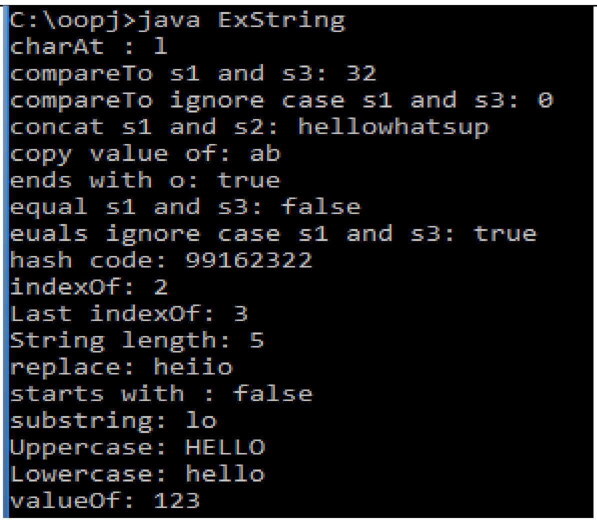
30) **static String valueOf(primitive data type x)** :returns String conversion of primitive data value.

Example,

```
public class ExString {
      public static void main( String args [])
      {
      String s1 = "hello";
      String s2 = "whatsup";
      String s3 = new String( "Hello" );
      char[] s4  = { 'a' , 'b' };
      System.out.println( "charAt : " + s1.charAt(2));
      System.out.println( "compareTo s1 and s3: " + s1.compareTo(s3));
      System.out.println( "compareTo ignore case s1 and s3: " +
                              s1.compareToIgnoreCase(s3));
      System.out.println( "concat s1 and s2: " + s1.concat(s2));
      System.out.println( "copy value of: " + String.copyValueOf(s4));
      System.out.println( "ends with o: " + s1.endsWith("o"));
      System.out.println( "equal s1 and s3: " + s1.equals(s3));
      System.out.println( "euals ignore case s1 and s3: " + s1.equalsIgnoreCase(s3));
      byte[] b = s1.getBytes();
```

```
        System.out.println( "hash code: " + s1.hashCode());
        System.out.println( "indexOf: " + s1.indexOf('l'));
        System.out.println( "Last indexOf: " + s1.lastIndexOf('l'));
        System.out.println( "String length: " + s1.length());
        System.out.println( "replace: " + s1.replace('l','i'));
        System.out.println( "starts with : " + s1.endsWith("h"));
        System.out.println( "substring: " + s1.substring(3));
        char ar1 [] = s1.toCharArray();
        System.out.println(" Uppercase: " + s1.toUpperCase());
        System.out.println(" Lowercase: " + s1.toLowerCase());
        System.out.println(" valueOf: " + String.valueOf(123));

        }
}
```



```
C:\oopj>java ExString
charAt : l
compareTo s1 and s3: 32
compareTo ignore case s1 and s3: 0
concat s1 and s2: hellowhatsup
copy value of: ab
ends with o: true
equal s1 and s3: false
euals ignore case s1 and s3: true
hash code: 99162322
indexOf: 2
Last indexOf: 3
String length: 5
replace: heiio
starts with : false
substring: lo
Uppercase: HELLO
Lowercase: hello
valueOf: 123
```

Figure-49 Output of program

4.13 STRINGBUFFER CLASS

StringBuffer is also a class of java.lang package. It is also used to create and manipulate the strings in java. The StringBuffer is used to create mutable strings.

We can create StringBuffer using following constructors,

StringBuffer() : creates an empty string buffer with the initial capacity of 16.
StringBuffer(String str) : creates a string buffer with the specified string.

StringBuffer(int capacity) : creates an empty string buffer with the specified capacity as length.

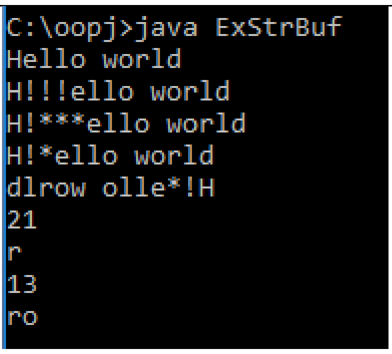
1) **StringBuffer append(String s)**: is used to append the specified string with this string.
2) **StringBuffer insert(int offset, String s)**: is used to insert a string with this string at the specified position.
3) **StringBuffer replace(int startIndex, int endIndex, String str)**: is used to replace the string from specified startIndex and endIndex.
4) **StringBuffer delete(int startIndex, int endIndex)**: is used to delete the string from specified startIndex and endIndex.
5) **StringBuffer reverse():** is used to reverse the string.
6) **int capacity()**: is used to return the current capacity.
7) **char charAt(int index)**: is used to return the character at the specified position.
8) **int length()**: is used to return the length of the string i.e. total number of characters.
9) **String substring(int beginIndex)**: is used to return the substring from the specified beginIndex.
10) **String substring(int beginIndex, int endIndex)**: is used to return the substring from the specified beginIndex and endIndex.

Example,

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

        StringBuffer s1 = new StringBuffer("Hello");
        s1.append( " world" );
        System.out.println( s1 );
        s1.insert ( 1 , "!!!");
        System.out.println( s1 );
        s1.replace ( 2, 4, "***" );
        System.out.println( s1 );
        s1.delete( 2, 4);
        System.out.println( s1 );
        s1.reverse();
        System.out.println( s1 );
        System.out.println( s1.capacity() );
        System.out.println( s1.charAt(2) );
        System.out.println( s1.length() );
        System.out.println( s1.substring(2,4) );

        }
}
```



```
C:\oopj>java ExStrBuf
Hello world
H!!!ello world
H!***ello world
H!*ello world
dlrow olle*!H
21
r
13
ro
```

Figure-50 Output of program

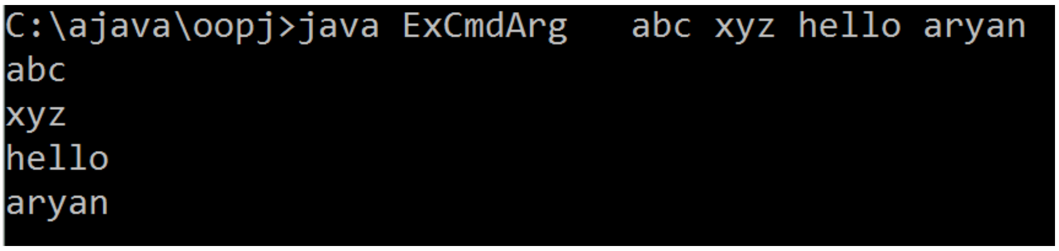
4.14 COMMAND LINE ARGUMENTS

Command line arguments are the arguments pass to java program when we run it. They are always in form of String. We can pass one or more string separated by space while running java program using java.exe. The java program accepts those strings in a String array as a parameter of main method. These arguments passed from the console to main method and can be received in the java program. They can be used as an input.

For example,

In this example, the command line arguments stored inside the array args[] and can be used inside the program.

```
public class ExCmdArg{
        public static void main(String args[])
        {
        for( int j = 0; j < args.length ; j++)
         System.out.println(args[j]);
        }
}
```



```
C:\ajava\oopj>java ExCmdArg   abc xyz hello aryan
abc
xyz
hello
aryan
```

Figure-51 Output of program

Here, the strings “abc”, “xyz”, “hello” and “aryan” are command line arguments**.**

4.15 GENERIC IN JAVA

Java Generics were introduced in JDK 5.0 with the aim of reducing bugs and adding an extra layer of abstraction over types. Generics in Java is similar to templates in C++. The idea is to allow type (user defined types) to be a parameter to methods, classes and interfaces. For example, classes like HashSet, ArrayList, HashMap, etc use generics very well. We can use them for any type.

Like C++, we use <> to specify parameter types in generic class creation. To create objects of generic class, we use following syntax.

BaseType <Type> obj = new BaseType <Type>()

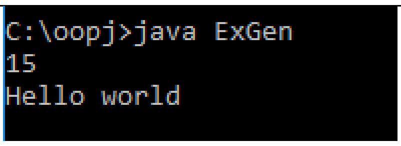
For example,

```
class Test<T>          // generic class
{
   T obj;
   Test(T obj) {  this.obj = obj;  }
   public T getObject()  { return this.obj; }
}

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

       Test <Integer> Obj1 =  new Test<Integer>(15);
       System.out.println( Obj1.getObject() );

        Test <String> Obj2 =  new Test<String>( " Hello world " );
       System.out.println( Obj2.getObject() );
   }
}
```



```
C:\oopj>java ExGen
15
Hello world
```

Figure-52 Output of program

4.16 LET US SUM UP

Access modifier: They are the key words which are use to restrict access of class member variables and methods.

public: This key word allows access of member functions and methods of class everywhere outside the class.

private: This key word allows access of member functions and methods of class only inside the class in which they are declared.

protected: This key word allows access of member functions and methods of class inside the class in which they are declared and in subclass of the class.

default: This key word allows access of member functions and methods of class only inside the classes of the package in which class resides.

recursion: It is a call of the function in itself**.**

nested class: We can create a class as a member of the class. This concept is called nested class.

outer class and inner class: In nested class, the class in which the member class is defined is called outer class. And the member class is called inner class.

String class: Strings are the non mutable sequence of characters.

StringBuffer class: They are mutable sequence of characters.

Command line arguments: They can be used to input in java program while running them on command prompt.

Generic: The idea of Generic is to allow type to be a parameter to methods, classes and interfaces like template of C++.

4.17 CHECK YOUR PROGRESS

- True-False with reason.

1. The recursion is calling a member function of a class into other member function.
2. Command line arguments can be used to give input to program.
3. Nested class is defining more than one class in same java program file.
4. We can not declare a class static.
5. Private member of the class can be accessed outside the class which is subclass.
6. Protected members of the class can be accessed inside the class in which they are declared.
7. Public members of a class can be accessed from everywhere.
8. For default access modifier the friendly keyword is used.
9. We can only pass strings as a command line arguments.
10. String is non mutable series of characters.

➢ MCQ.

1) The output of the following fraction of code is

```
public class Test{
        public static void main(String args[]){
              String s1 = new String("Hello");
                String s2 = new String("Hellow");
                System.out.println(s1 = s2);
      }
}
```

a. Hello   c. Compilation error

b. Hellow   d. Throws an exception

2) What will be the output of the following program code?

```
class LogicalCompare{
        public static void main(String args[]){
                String str1 = new String("OKAY");
                String str2 = new String(str1);
                System.out.println(str1 == str2);
        }
}
```

a.true   c.0

b.false   d.1

3) What will be the output of the following program?

```
public class Test{
        public static void main(String args[]){
                String s1 = "java";
                String s2 = "java";
                System.out.println(s1.equals(s2));
                System.out.println(s1 == s2);
        }
}
```

a.false true   c.true false

b.false false   d.true true

4) Determine output:

```
public class Test{
        public static void main(String args[]){
                String s1 = "SITHA";
                String s2 = "RAMA";
                System.out.println(s1.charAt(0) > s2.charAt(0));
        }
}
```

a.true c.0

b.false d.Compilation error

5) toString() method is defined in

a. java.lang.String c. java.lang.util

b. java.lang.Object d. None of these

6) The String method compareTo() returns

a. true c. an int value

b. false d. 1

7) What will be the output?

```
String str1 = "abcde";
System.out.println(str1.substring(1, 3));
```

a abc c. bcd

b. bc d. abcd

8) What is the output of the following println statement?

```
String str1 = "Hellow";
System.out.println(str1.indexOf('t'));
```

a. true c. 1

b. false d. -1

9) What will be the output of the following program?

```
public class Test{
        public static void main(String args[]){
                String str1 = "one";
                String str2 = "two";
                System.out.println(str1.concat(str2));
        }
}
```

a. one
b. two
c. onetwo
d. twoone
e. None of these

10)String str1 = "Kolkata".replace('k', 'a');

In the above statement, the effect on string Kolkata is

a. The first occurrence of k is replaced by a.
b. All characters k are replaced by a.
c. All characters a are replaced by k.
d. Displays error message

11)Which statement, if placed in a class other than MyOuter or MyInner, instantiates an instance of the nested class?

```
public class MyOuter {
    public static class MyInner
    {
        public static void foo() { }
    }
}
```

a. MyOuter.MyInner m = new MyOuter.MyInner();
b. MyOuter.MyInner mi = new MyInner();
c. MyOuter m = new MyOuter();
   MyOuter.MyInner mi = m.new MyOuter.MyInner();
d. MyInner mi = new MyOuter.MyInner();

12)Which statement, inserted at line 10, creates an instance of Bar?

```
class Foo
{
    class Bar{ }
}
class Test
{
    public static void main (String [] args)
```

```
        {
            Foo f = new Foo();
            /* Line 10: Missing statement ? */
        }
    }
```

a) Foo.Bar b = new Foo.Bar();　　　　　c) Bar b = new f.Bar();

b) Foo.Bar b = f.new Bar();　　　　　d) Bar b = f.new Bar();

13)Which constructs an anonymous inner class instance?

a) Runnable r = new Runnable() { };

b) Runnable r = new Runnable(public void run() { });

c) Runnable r = new Runnable { public void run(){}};

d) System.out.println(new Runnable() {public void run() { }});

14)What will be the output of the program?

```
public abstract class AbstractTest
{
    public int getNum()
    {
        return 45;
    }
    public abstract class Bar
    {
        public int getNum()
        {
            return 38;
        }
    }
    public static void main (String [] args)
    {
        AbstractTest t = new AbstractTest()
        {
            public int getNum()
            {
                return 22;
            }
        };
        AbstractTest.Bar f = t.new Bar()
        {
            public int getNum()
            {
                return 57;
            }
        };

        System.out.println(f.getNum() + " " + t.getNum());
    }
}
```

a) 57 22 c) 45 57

b) 45 38 d) An exception occurs at runtime.

15)Which statement is true about a static nested class?

a) You must have a reference to an instance of the enclosing class in order to instantiate it.

b) It does not have access to nonstatic members of the enclosing class.

c) It's variables and methods must be static.

d) It must extend the enclosing class.

16)What will be the output of the program?

```
public class TestObj
{
    public static void main (String [] args)
    {
        Object o = new Object() /* Line 5 */
        {
            public boolean equals(Object obj)
            {
                return true;
            }
        }     /* Line 11 */

        System.out.println(o.equals("Fred"));
```

```
   }
}
```

a) It prints "true".

b) It prints "Fred".

c) An exception occurs at runtime.

d) Compilation fails

17)What is Recursion in Java?

a) Recursion is a class

b) Recursion is a process of defining a method that calls other methods repeatedly

c) Recursion is a process of defining a method that calls itself repeatedly

d) Recursion is a process of defining a method that calls other methods which in turn call again this method

18)Which of these data types is used by operating system to manage the Recursion in Java?

a) Array

b) Stack

c) Queue

d) Tree

19) Which type of variable or method can ONLY be used within the current package?

a) Protected

b) Private

c) Public

d) Void

20)What is the output of this program?

```
class recursion
{
   int func (int n)
   {
      int result;
      result = func (n - 1);
      return result;
   }
}
```

```
class Output
{
    public static void main(String args[])
    {
        recursion obj = new recursion() ;
        System.out.print(obj.func(12));
    }
}
```

a) 0 c) Compilation Error
b) 1 d) Runtime Error

21)What is the output of this program?

```
class recursion
{
    int fact(int n)
    {
        int result;
        if (n == 1)
            return 1;
        result = fact(n - 1) * n;
        return result;
    }
}
class Output
{
    public static void main(String args[])  {
        recursion obj = new recursion() ;
        System.out.print(obj.fact(5));
    }
}
```

a) 24 c) 120
b) 30 d) 720

22)You have the following code in a file called Test.java

```
class Base{
     public static void main(String[] args){
        System.out.println("Hello");
     }
}
public class Test extends Base{}
```

What will happen if you try to compile and run this?

a. It will fail to compile.

b. Runtime error

c. Compiles and runs with no output.

d. Compiles and runs printing

23)Examine the following code. Where can the program use the variable fte?

```
public class Employee_Public_View {
public String employeeName = new String ();
public int jobCode;
private float fte;
float getFTE( float fte ) {
this.fte = 1;
return fte;
 }
}
```

a) In all classes within the program

b) Only in the Employee_Public_View class

c) It cannot be used

d) In the main function

24)Examine the following code. What is true about the variables and methods within the class NYCustomer?

```
class NYCustomer{
public long customerPhone;
public void getCustomerPhone() {
 }
}
```

a) They are private

b) They can be accessed by other packages and classes

c) They can be accessed only in subclasses

d) They are protected

4.18 CHECK YOUR PROGRESS: POSSIBLE ANSWERS

➢ True-False with reason

1. False. The recursion is calling a function of a class into the function itself.
2. True
3. False. Nested class is defining a class inside a class.
4. False. We can declare inner class of nested class static.
5. False. Private members of a class can be accessed inside a class only.
6. False. Protected members of the class can be accessed inside the class in which they are declared and inside the subclass.
7. True.
8. False. No keyword is used for default access.
9. True.
10. True.

➢ MCQ.

1) b	9) c	17) c
2)b	10) b	18) b
3) c	11) c	19) d
4) a	12) b	20) c
5) b	13) d	21) d
6) c	14) a	22) b
7) b	15) d	23) d
8) d	16) a	24) b

4.19 FURTHER READING

1) "Java 2: The Complete Reference" by Herbert Schildt, McGraw Hill Publications.
2) "Effective Java" by Joshua Bloch, Pearson Education
3) Nested classes in Java | Core Java Tutorial' | Studytonight https://www.studytonight.com/ java/ nested-classes.php

4) What Is Recursion in Java Programming ? - dummies
5) https:// www.dummies.com/ programming/ java/ what-is-recursion-in-java-programming/

4.20 ASSIGNMENTS

➢ Write java program for following

1) Print Fibonacci series up to n elements using recursion.
2) Implement binary search using recursion.
3) Create a class Student with attributes roll number, name, address, phone numbers. To store address, create an Address class. An Address class can have PhoneNumbers inner class which holds all the phone numbers associated with an address and may have some extra functionality, like returning the best phone number (the most used one). All classes have get methods and print methods to input and print the data values respectively.
4) Find GCD of a number using recursion.
5) Create a class Customer with properties customer ID, name, address, phone number, date of birth and function to get and print these attributes. Inherit the class Account from Customer class with account number, account type, rate of interest and balance properties and functions to get and print these properties. Also in account class implement the deposit and withdraw function. Use appropriate access modifier with attributes and methods of each class.
6) Implement the above example considering customer as an abstract class which is inherited as Account class. Also inherit the Loan class from the Customer class which has properties like loan number, rate of interest, loan duration, loan amount, date of installment etc and methods to get and print value of these attributes. The Loan class also has method to pay installment. Use appropriate access modifier with attributes and methods of each class. After implementation of classes show their use in main method.
7) For educational institute design an application in which a Person with person id, name, address, department and phone number can be a faculty or a student. A faculty can have other attributes like degree, designation, specialization, experience etc. A student can have attributes like semester;

results etc. create appropriate classes and methods in the classes. Use appropriate access modifier with attributes and methods of each class. Also show their use in main method.

8) Implement a java program to input a String from command line argument and convert it into upper case without using toUpperCase function.

9) Implement a MyString class with your own reverse, getBytes and parseInt function in it. (Do not use readymade functions available in String class).

10) To input a paragraph from console and convert word at even position into upper case.

11) To input a paragraph from console and replace a word “is” with “are” in the input paragraph.