Assignment - 2

Name :- Tanishq Chauhan

Roll No :- 21C3184

Subject :- Object Oriented Programming (CER3C2)

Semester :- 3rd Semester

Branch :- Computer Science & Engineering (CS-B)

Chauhan

Tanishq Chauhan
21C3184
CS-B

Chauhan
Date
Page 01

Object Oriented Programming (CER3C2)

Assignment - 2

Q.1. How many interface can a class inherit and can a same interface be inherited by two different classes? Explain it.

Answer of Q. No. 1.

An interface in Java is a blueprint of a class.
An interface is a reference type in Java. It is similar to class.
It is a collection of abstract methods.
A class implements an interface, thereby inheriting the abstract methods of the interface.

When a class implements an interface, you can think of the class as signing a contract, agreeing to perform the specific behaviors of the interface.
If a class does not perform all the behaviors of the interface, the class must declare itself as abstract.
A class uses the **implements** keyword to implement an interface.
When implementation interfaces, there are several rules.

- A class can implement more than one interface at a time.
- A class can extend only one class, but implement many interfaces.
- An interface can extend another interface, in a similar way as a class can extend another class.

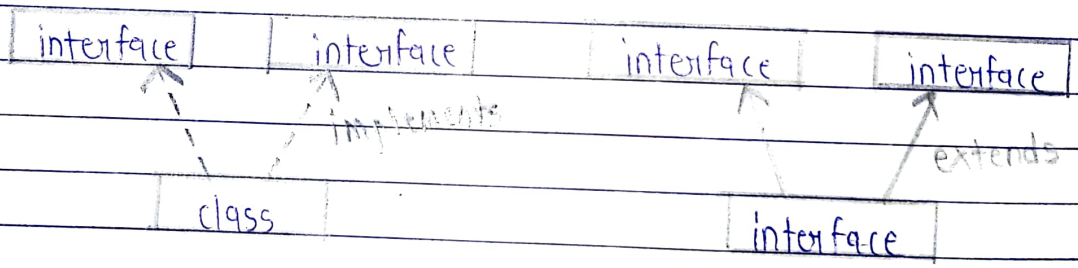
Tanishq Chauhan
21C3184
CS-B

Chauhan

Date
Page 02

Interfaces provide an alternative to multiple inheritance. Java programming language does not support multiple inheritance. But interface provide a good solution. Any class can implement a particular interface and importantly the interfaces are not a part of class hierarchy. So, the general rule is **extend one but implement many.**

Yes, a same interface be inherited by two different classes.
If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance.



Example

```
interface Printable
{
    void print();
}
interface Showable
{
    void show();
}
```

Tanishq Chauhan
21C3184
CS-B


```
class A implements Printable, Showable
{
  public void print()
  {
    System.out.println("Hello");
  }

  public void show()
  {
    System.out.println("Welcome");
  }

  public static void main(String args[])
  {
    A obj = new A();
    obj.print();
    obj.show();
  }
}
```

Output

```
Hello
Welcome
```

Tanishq Chauhan
21C3184
CS-B

Chauhan
Date
Page 04

Q.2 What is the difference between a Package and a Class? Mention the use of various packages of Java API.

Answer of Q.No.2

Difference Between Package & Class

In Java. Packages are used to organize classes belonging to the same functionality and Java packages can be stored in compressed file called .JAR files.
Class is an entity that determines how an object will behave and what the object will contain.

Package is a collection of classes and class is a collection of objects.
A class is a declaration (and often implementation) which encapsulates the member properties and methods which instances of that type will posses. You can create instances of a class as distinct, individual objects that behave as specified.
A package is simply a namespace under which you can categorize classes, analogous to a "folder" or "path", so you can group classes with related purpose or functionality.

Quite simply a java package is a namespace where classes are organized based on the same category.
Java classes is an object that consists of behaviours (methods), properties (variables), constructors (if any, for object creation) etc. that exists inside a folder (package).

Tanishq Chauhan
21C3184
CS-B

Uses of Various Packages

We use package to write a better, maintable code. Packages are divided into two categories.

1. Built-in Packages

These packages consist of a large number of classes which are a part of Java API.
Some of the commonly used built-in packages are:-

(i) java.lang : It contains language support classes (e.g. classed which defines primitive data type, math operations). This package is automatically imported.

(ii) java.io : It contains classed for supporting input/output operations.

(iii) java.util : It contains utility classes which implement data structure like linked list, Dictionary and support; for Date/Time operations.

(iv) java.applet : Contain classes for creating Applets.

(v) java.awt : It contain classes for implementing the components for graphical user interfaces (like button, menus etc).

(vi) java.net : It contain classes for supporting networking operations.

Tanishq Chauhan
21C3184
CS-B

2. User-defined Packages

These are the packages that are defined by the user. First we create a directory myPackage (name should be same as the name of the package).
Then create the MyClass inside the directory with the first statement being the package names.

Tanishq Chauhan
21C3184
CS-B

Chauhan
Date
Page 07

Q.3. What is meant by abstract data type? Compare abstract data type with existing data type. Explain the significance of it in object oriented programming.

Answer of Q. No. 3

Abstract Data Type

- Abstract Data Type (ADT) is a type (or class) for objects whose behaviour is defined by a set of value and a set of operations.
- The definition of ADT only mentions what operations are to be performed but not how these operations will be implemented. It does not specify how data will be organized in memory and what algorithms will be used for implementing the operations.
- It is called "abstract" because it gives an implementation independent view.
- The process of providing only the essentials and hiding the details is known as abstraction.

Compare Abstract Data Type with Existing Data Type

Data type specify the different sizes and values that can be stored in the variable, these are the types of data types in Java:

1) Primitive Data Type :- It includes boolean, char, byte, short, int, long, float and double.
2) Non-Primitive Data Type :- The non-primitive data type includes, classes, interfaces & arrays.

Tanishq Chauhan
2103184
CS-B

Chauhan

Date
Page 08

- Abstract data type is abstraction that define set of values and set of operation on these values.
- It is user define data type.
- ADT is made of with primitive data types.
- It is a type (or class) for objects whose behaviour is defined by a set of value and set of operations.
- Abstract data type is not necessarily an OPP concept.
- Common example include list, stacks sets etc.
- It allocate memory when data is stored

Significance of Abstract Data Type

Abstract data type is important for large scale programming, they package data structure and operations on them, hiding internal details.

Abstract data type is a important part of object oriented programming. an ADT implemented specific data type or data structure in many programming languages in a formal specification it often implemented as modules or method declaring procedures of how an interface of operation used, sometimes with comments defining the constraint.

ADT used in like List, Set, Multiset, Stack, Queue Priority Queue, Map, Multi-Map, Graph, Tree, Double ended queue.

Tanishq Chauhan
2103184
CS-B

Chauhan
Date
Page 09

Q.4. What is the requirement of inter-thread communication? Explain the use of wait(), notify() and notifyAll() methods for inter-thread communication.

Answer of Q. No. 4

Requirement of Inter-Thread Communication

- Inter-thread communication or co-operation is all about allowing synchronized threads to communicate with each other.
- To avoid polling, Java includes an elegant interprocess communication mechanism via the wait(), notify(), and notifyAll() methods.
- These methods are implemented as final methods in Object, so all classes have them.
- All three methods can be called only from within a synchronized context.
- Although conceptually advanced from a computer science perspective, the rules for the using these methods are actually quite simple:

(i) wait() method

- The wait() method causes current thread to release the lock and wait until either another thread invokes the notify() method or the notifyAll() method for this object, or a specified amount of time has elapsed.

Tanishq Chauhan
21C3184
CS-B

@Chauhan

Date
Page 10

- The current thread must own this object's monitor, so it must be called from the synchronized method only otherwise it will throw exception.

Method	Description
public final void wait() throws InterruptedException	It waits until object is notified.
public final void wait (long timeout) throws InterruptedException	It waits for the specified amount of time.

(ii) notify() method

- The notify() method wakes up a single thread that is waiting on this object's monitor.
- If any threads are waiting on this object, one of them is chosen to be awakened.
- The choice is arbitrary and occurs at the discretion of the implementation.

Syntax : public final void notify()

(iii) notifyAll() method

- It wakes up all the threads called wait() on the same object.

Syntax : public final void notifyAll()

Tanishq Chauhan
21C3184
CS-B

Q.5. Discuss how exceptions are handled in Java. What is the role of finally block.

Answer of Q. No. 5

The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that the normal flow of the application can be maintained.

In Java, an exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

Exception handling is a mechanism to handle runtime errors such as ClassNOTFoundException, IOException, SQLException, RemoteException etc.

Whenever inside a method, if an exception has occurred, the method creates an Object known as Exception Object and hands it off to the run-time system (JVM). The exception object contains name and description of the exception, and current state of the program where exception has occurred.
Creating the Exception Object and handling it to the run-time system is called throwing an Exception.
There might be the list of methods that had been called to get to the method where exception was occurred.
This ordered list of the methods is called Call Stack.

Tanishq Chauhan
21C3184
CS-B

The run-time system searches the call stack to find the method that contains block of code that can handle the occurred exception. The block of the code is called Exception handler.
The run-time system starts searching from the method in which exception occurred, proceeds through call stack in the reverse order in which methods were called.
If it finds appropriate handler then it passes the occurred exception to it. Appropriate handler means the type of the exception object thrown matches the type of the exception object it can handle.
If run-time system searches all the methods on call stack and couldn't have found the appropriate handler then run-time system handover the Exception Object to default exception handler, which is part of run-time system. This handler prints the exception information in the following format and terminates program abnormally.

Exception in thread "xxx" Name of Exception : Description
... . // Call Stack

Example

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

```
{
    try {
        //code that may raise exception
        int data = 100/0;
    }
    catch (ArithmeticException e)
    {
        System.out.println(e);
    }
    System.out.println("Rest code ...");
}
}
```

Output

```
java.lang.ArithmeticException : / by zero
Rest code ...
```

Finally Block

- Java finally block is block used to execute important code such as closing the connection etc.
- Java Finally block is always executed whether an exception is handled or not. Therefore, it contains all the necessary statements that need to be printed regardless of the exception occurs or not.

Tanishq Chauhan
2103184
CS-B

- The finally block follows the try-catch block.
- Finally block in Java can be used to put "cleanup" code such as closing a file, closing connection etc.
- The important statements to be printed can be placed in the finally block.
- Java Finally block can be used in different cases.

Case 1 : When an exception does not occur
Case 2 : When an exception occur but not handled by the catch block.
Case 3 : When an exception occurs and is handled by the catch block

Example :

```
class FinallyBlock
{
   public static void main (String args[])
   {
      try{
         //below code do not throw any exception
          int data = 25/5;
          System.out.println(data);
        }

      // Catch won't be executed

      catch (NullPointerException e) {
                                  System.out.println(e); }
```

Tanishq Chauhan
21C3184
CS-B


```
// executed regardless of exception occurred or not
finally {
        System.out.println("Finally Block is always
                                executed");
        }

System.out.println("Rest code....");
    }
}
```

Output

```
5
Finally Block is always executed
Rest code....
```

Tanishq Chauhan
21C3184
CS-B

Q.6. Create a try block that is likely to generate three types of exception and then incorporate necessary catch blocks to catch and handle them appropriately.

Answer of Q. No. 6.

```
class Nesting
{
   public static void main (String args[])
   {     //main try-block
      try
      {
         // try-block 2
          try
          {
             // try-block 3
              try
              {
                 int arr[] = {1,2,3,4,5,6,7};
                 System.out.println (arr[10]);
              }

             // handles ArithmeticException if any

             catch (ArithmeticException e)
             {
                System.out.println("Arithmetic Exception");
                System.out.println (" try-block 1");
             }}
```

Tanishq Chauhan
2103184
CS-B


```
// handles ArithmeticException if any
catch (ArithmeticException e)
{
    System.out.println ("Arithmetic Exception");
    System.out.println ("try-block 2");
}
}

// handles ArrayIndexOutOfBoundException if any
catch (ArrayIndexOutOfBoundsException e4)
{
    System.out.println ("ArrayIndexOutOfBoundsException");
    System.out.println ("main try-block");
}

catch (Exception e5)
{
    System.out.print ("Exception");
    System.out.println ("Handled in main
                         try-block");
}
}
}
```

Tanisha Chauhan
21C3184
CS-B

Chauhan

Date
Page 18

Q-7. Create a package myNum which contains classes "PerfectNumber" and "PrimeNumber" that performs functionality as their name. Write main method within another class which uses functionality of both the classes.

Answer of Q. No. 7

```
Package myNum;
import java.util.*;

Public class PerfectNumber

{

    int sum = 0;
    int i = 1;
    Scanner sc = new Scanner (System.in);
    int n = sc.nextInt();

    public void M2()
    {
        while (i<=n/2)
        {
            if (n%i==0)
            {
                sum = sum + i;
            }
            i++;
        }
```

```
        if (sum == n)
        {
            System.out.println(" Number is Perfect: " +n);
        }
        else
        {
            System.out.println(" Number is not Perfect: " +n);
        }
    }
}
```

```
package myNum;
import java.util.*;

public class PrimeNumber
{
    boolean flag = false;
    Scanner sc = new Scanner(System.in);
    int n = sc.nextInt();

    public void M1()
    {
        for (int i=2 ; i<=n; ++i)
        {
            if (n%i==0)
            {
                flag = true;
                break;
```

Tanisha Chauhan
21C3184
CS-B


```
        }
      }
      if (!flag)
      {
        System.out.println(" Prime Number: "+n);

      else
      {
        System.out.println(" Not a Prime Number: "+n);
      }
    }
  }
```

```
import myNum.*;
public class Test
{
  public static void main (String args[])
  {
    PrimeNumber pn = new PrimeNumber();
    PerfectNumber pft = new PerfectNumber();

    pn.m1();
    pft.m2();
  }
}
```

Output

```
6
6
Not a Prime Number: 6
Number is Perfect: 6
```

Tanishq Chauhan
21C3184
CS-B

Chauhan

Date
Page 21

Q.8. What are the scope and lifetime of variables in java.

Answer of Q. No. 8

Scope of a variable refers to in which areas or section of a program can the variable be accessed and lifetime of a variable refers to how long the variable stays alive in memory.

As we know there are three types of variables.

1. Instance Variables

- A variable which is declared inside a class and outside all the methods and blocks is an instance variable.
- The general scope of an instance variable is throughout the class except in static methods.
- The lifetime of an instance variable is untill the object stays in memory.

Example

```
class Instance
{
    int x,4;
    static int result;
```

Tanishq Chauhan
21C3184
CS-B


```
void add (int a, int b)
{
    x = a;
    y = b;
    int sum = x + y;
    System.out.println ("Sum = " + sum);
}

public static void main (String args[])
{
    Instance obj = new Instance ();
    obj.add (10, 20);
}
}
```

Output

Sum = 30

2. Class Variables

- A variable which is declared inside a class, outside all the blocks and is marked static is known as a class variable.
- The general scope of a class variable is throughout the class and the lifetime of a class variable is until the end of the program or as long as the class is loaded in memory.

Tanishq Chauhan
2103184
CS-B

Example

```
class sample
{
    int x, y;
    static int result;

    void add(int a , int b)
    {
        x = a;
        y = b;
        int sum = x+y;
        System.out.println ("Sum = "+ sum);
    }

    Public static void main (String args[])
    {
        sample obj = new sample();
        obj.add (10,20);
    }
}
```

Output

Sum = 30

Tanishq Chauhan
21C3184
CS-B

Chauhan

Date
Page 24

3. Local Variables

- All other variables which are not instance and class variables are treated as local variables including the parameters in a method.
- Scope of a local variable is within the block in which it is declared and the lifetime of a local variable is until the control leaves the block in which it is declared.

Example

```
class local
{
    int x,y;
    static int result;

    void add (int a , int b)
    {
        x = a;
        y = b;
        int sum = x + y;
        System.out.println ("Sum = " + sum);
    }

public static void main (String args[])
{
    local
    ~~Sample~~ obj = new ~~sample~~ local();
    obj.add (30, 20); } }
```

Output

Sum = 50

Tanishq Chauhan
21C3184
CS-B

tChauhan

Date
Page 25

Q9. Write complete java program to define a 10 user defined exception named "FewArgumentsException". This exception should be thrown by program if it receives lessthan three command line arguments. On catching this exception your program should set first three command line arguments with string "a", "b" and "c". In any case i.e. user defined exception thrown or not, program should print all the arguments. Your program should be safe from NullPointerException or ArrayIndexOutOfBoundException.

Answer of Q. No. 9

```
import java.util.*;
import java.io.IOException;
class FewArgumentsException extends RuntimeException
{
    public FewArgumentsException()
    {
        super("Command line args");
    }

    public FewArgumentsException(String args)
    {
        super(args);
    }
}
```

Tanishq Chauhan
2103184
CS-B

Chauhan

Date
Page 26

```
public class extends FewArgumentsException
{
  public static void main (String args[])
  {
    Scanner sc = new Scanner (System.in);
    if (args.length <3)
    {
      throw new FewArgumentsException(" Args are
                  less than 3");
    }
    else
    {
      for (int i=0 ; i<args.length ; i++)
      {
        System.out.println (args[i]);
      }
    }
  }
}
```