National University of Computer & Emerging Sciences,
Karachi
Computer Science Department
Fall 2022, Lab Manual – 01

Course Code: CL-2001	Course : Data Structures - Lab
Instructor(s) :	Abeer Gauher, Sobia Iftikhar

LAB - 1

INTRODUCTION TO DATA STRUCTURES AND THEIR IMPLEMENTATION

Reading and Writing from files

Reading:

The BufferedReader class in Java provides a convenient way to read text from a character-input stream. Its read() method can be used to read a specified number of characters or an array of characters. It can significantly improve the performance of your program by reading data in larger chunks.

What is BufferedReader in Java?

BufferReader is a Java class used to read characters, arrays, and lines from an input stream. It is similar to a FileReader but provides buffering functionality as well.

BufferReader allows fast reading by buffering data, and provides several methods for reading lines of text. This allows you to read large amounts of text more efficiently than if you were reading it directly from the stream.

How to Program BufferedReader in Java

- To use BufferedReader, import the java.io.BufferedReader package.
- Then, you can create a BufferedReader object by passing a FileReader object to its constructor.
- Next, you can call the readLine() method of the BufferedReader class to read a line of text from the file.

Example:

```
package com.company;
import java.io.*;

public class lab1  {
    public static void main(String[] args) {
        try {
            FileReader fr = new FileReader( fileName: "Demo.txt");
            BufferedReader bufferedReader = new BufferedReader(fr);
            String line;
            while((line=bufferedReader.readLine()) != null){
            System.out.println(line);}
            bufferedReader.close();
        } catch (IOException e) {
            System.out.println("Error while retrieving data…");
        }
    }
}
```

Writing:

The BufferedWriter class of the java.io package can be used with other writers to write data (in characters) more efficiently.It extends the abstract class Writer.

Working of BufferedWriter

The BufferedWriter maintains an internal buffer.During the write operation, the characters are written to the internal buffer instead of the disk. Once the buffer is filled or the writer is closed, the whole characters in the buffer are written to the disk.

Example:

```java
package com.company;
import java.io.*;

public class lab1 {

    public static void main(String args[]) {

        String data = "This is the data in the output file";

        try {
            FileWriter file = new FileWriter( fileName: "output.txt");
            BufferedWriter output = new BufferedWriter(file);
            output.write(data);
            output.close();
        }

        catch (Exception e) {
            e.getStackTrace();
        }
    }
}
```

Garbage Collection and “new” keyword

“new” keyword:

The Java new keyword is used to create an instance of the class. In other words, it instantiates a class by allocating memory for a new object and returning a reference to that memory. We can also use the new keyword to create the array object.

Syntax

NewExample obj=new NewExample();

Points to remember:

- It is used to create the object.
- It allocates the memory at runtime.
- All objects occupy memory in the heap area.
- It invokes the object constructor.

Example:

```
package com.company;
import java.io.*;

public class lab1 {

    void display()
    {
        System.out.println("Invoking Method");
    }

    public static void main(String[] args) {
        lab1 obj=new lab1();
        obj.display();
    }

}
```

Garbage Collection in JAVA:

A Definition of Java Garbage Collection

Java garbage collection is the process by which Java programs perform automatic memory management.When Java programs run on the JVM, objects are created on the heap. Eventually, some objects will no longer be needed. The garbage collector finds these unused objects and deletes them to free up memory.

How Java Garbage Collection Works

Java garbage collection is an automatic process. The programmer does not need to explicitly mark objects to be deleted.

Benefits of Java Garbage Collection

The benefit of Java garbage collection is that it automatically handles the deletion of unused objects to free up memory resources. Programmers working in languages without garbage collection (like C and C++) must implement manual memory management in their code.

Way for requesting JVM to run Garbage Collector

We can request JVM to run Garbage Collector.

- Using System.gc() method: System class contain static method gc() for requesting JVM to run Garbage Collector.

Finalization

Just before destroying an object, Garbage Collector calls finalize() method on the object to perform cleanup activities.
finalize() method is present in Object class with the following prototype.

protected void finalize() throws Throwable

Object class finalize() method has an empty implementation. Thus, it is recommended to override the finalize() method to dispose of system resources.

Example:

```
package com.company;
import java.io.*;

public class lab1{
    public void finalize(){System.out.println("object is garbage collected");}
    public static void main(String args[]){
        lab1 s1=new lab1();
        lab1 s2=new lab1();
        s1=null;
        s2=null;
        System.gc();
    }
}
```

```
object is garbage collected
object is garbage collected
```

Shallow Copy and Deep Copy

Shallow Copy:

When we do a copy of some entity to create two or more than two entities such that changes in one entity are reflected in the other entities as well, then we can say we have done a shallow copy. In shallow copy, new memory allocation never happens for the other entities, and only reference is copied to the other entities.

Example:

```java
package com.company;
import java.io.*;

class shallow
{
    int x = 50;
}
public class lab1
{
    public static void main(String args[])
    {
        shallow obj1 = new shallow();

// it will copy the reference, not value
        shallow obj2 = obj1;

// updating the value to 25
        obj2.x = 25;
        System.out.println("The value of x is: " + obj1.x);
    }
}
```

```
The value of x is: 25
```

Deep Copy:

When we do a copy of some entity to create two or more than two entities such that changes in one entity are not reflected in the other entities, then we can say we have done a deep copy. In the deep copy, a new memory allocation happens for the other entities, and reference is not copied to the other entities.

Example:

```
package com.company;
import java.io.*;

class DeepCopyExample
{
    int x = 70;
}
public class lab1
{
    public static void main(String argvs[])
    {
        DeepCopyExample obj1 = new DeepCopyExample();
        DeepCopyExample obj2 = new DeepCopyExample();

// updating the value to 150
        obj2.x = 150;
        System.out.println("The value of x is: " + obj1.x);
    }
}
```

```
The value of x is: 70
```

Lab Tasks:

Task#1:
Create a .txt file that stores the details of a student.

- You are required to take the student's ID and name as input and write it to the file.
- Ask the user if he has completed Programming Fundamentals and Object Oriented Programming (Yes or No) and the obtained marks for both subjects should be 50 and above.
- If both conditions are satisfied, write "You can now register for Data Structures" or else write "You cannot register for Data structures".
- After writing all the information in the file, read and display the information.

Task#2:
Create a .txt file that stores details of employees at an office.

- Prompt the user whether the user is "Admin" or a "Staff".
- Write the employee name and Department in the file. If the user is a Admin ask the user then each hour is paid as Rs.1500. Calculate the monthly salary of the Admin and store all the details in the file.
- Write the Staff's name and Department in the file. Ask the user how many hours does he work in a week. Each hour is paid as Rs.950. Calculate the monthly salary of the Staff and store all the details in the file.
- After writing all the information in the file, read and display the information.

Task#3:
You are required to implement a program that manages the details of a hospital.

- Create a class named as Patient. The class has Patient's name, Patient ID and whether the Patient visits a General or a Special Doctor (can be a Boolean variable).
- Prompt the user whether he want to visit a General or Special Doctor.
- For General Doctors the fee calculation is as follows:
  Charge fee = Rs 1200, Service Tax = Rs.450
  Calculate the user's bill for the visit.
  If the user has previously visited the General Doctor then the user gets a 5% discount on the total bill.
- For Special Doctors the fee calculation is as follows:
  Charge fee = Rs 1800, Service Tax = Rs.650
  Calculate the user's bill for the visit.
  If the user has previously visited the General Doctor then the user gets a 7% discount on the total bill.
- Create two objects for Patient class, one for general and one for special.
- Then use garbage collection to delete those objects. Display "Object Deleted".

Task#4:
This task will help you understand the difference deep copy and shallow copy.

- Write a class called Holiday. This class has three instance variables: name, Day & month.

- Write a constructor for the class Holiday, which takes a String representing the name, an int representing the day, and a String representing the month as its arguments, and sets the class variables to these values.
- Write a method inSameMonth, which compares and returns the Boolean value true if they have the same month, and false if they do not.
- Creates a Holiday instance with the name “Independence Day”, with the day “14”, and with the month “August”.
- Create two more objects one using shallow copy and one using deep copy.
- Change the shallow copy details to “Defence Day” with the day “6”, and with the month “September”.
- Change the deep copy details to “Labour Day” with the day “1”, and with the month “May”.
- Display all the details of all the 3 objects after the changes.