ASSIGNMENT-12

1) Why collection framework in java

*Collection Framework is a powerful framework in java.
*This framework defines the most common methods that can be used for any collection of objects.
Benefits:
*Reduces programming effort
*Increases program speed and quality
*Allows interoperability among unrelated APIs
*Reduces effort to learn and to use new APIs
*Reduces effort to design new APIs

2)What is Collection interface?

*The Collection interface is a member of the Java Collections Framework. It is a part of java.util package. It is one of the root interfaces of the Collection Hierarchy.
*The Collection interface is not directly implemented by any class. However, it is implemented indirectly via its subtypes or subinterfaces like List, Queue, and Set.

3) what is package of collection framework?

*The Java collections framework provides a set of interfaces and classes to implement various data structures and algorithms.
*A collections framework is a unified architecture for representing and manipulating collections, enabling collections to be manipulated independently of implementation details.

4) which is root interface of Collection Framework?

*Java.util.package is the root interface of Collections Framework.

5)List the subinterface of Collection interface.

*List
*Set
*Queue

6)List out the classes which are implementing the List interface and Set Interface and Collection interface.

*ArrayList, Vector, LinkedList, PriorityQueue, HashSet, LinkedHashSet, TreeSet

8) Write a program to read and print the file properties like nameof the file, size,author,dateofcreation and dateofupdation using Properties class of collection framework.

```
package properties;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.Properties;

public class PropertiesPrg {

		public static void main(String[] args) {
			Properties p=new Properties();

			try {

				FileReader reader=new FileReader("D:\\
Assignment_12\\src\\properties\\file.properties");
				p.load(reader);

	System.out.println(p.getProperty("Name_of_the_file"));

	System.out.println(p.getProperty("Size_of_the_file"));
				System.out.println(p.getProperty("Author"));

	System.out.println(p.getProperty("Date_of_creation"));

	System.out.println(p.getProperty("Date_of_updation"));
				reader.close();

			}catch(FileNotFoundException  ex)
			{
				System.out.println(ex.getMessage());
			}catch(IOException  ex)
			{
				System.out.println(ex.getMessage());
			}
```

```
        }
}
```

9) Difference between the List and Set

*Set can hold only unique elements where as List can contain duplicate elements.
*Set is unordered while List is ordered; List maintains the order of object insertion.

10) Diffeence between ArrayList and LinkedList

* ArrayList internally uses a dynamic array to store the elements.LinkedList internally uses a doubly linked list to store the elements.
*Manipulation with ArrayList is slow.Manipulation with LinkedList is faster
*ArrayList is better for storing and accessing data.LinkedList is better for manipulating data.
*The memory location for the elements of an ArrayList is contiguous.The location for the elements of a linked list is not contagious.

11) Difference between HashMap and HashSet

*HashSet is an implementation of Set Interface which does not allow duplicate value. The main thing is, objects that are stored in HashSet must override equals() for check for equality, and hashCode() methods for no duplicate value are stored in our set.
*HashMap is an implementation of Map Interface, which maps a key to value. Duplicate keys are not allowed in a Map. Basically, Map Interface has two implementation classes HashMap and TreeMap the main difference is TreeMap maintains an order of the objects but HashMap will not. HashMap allows null values and null keys. Both HashSet and HashMap are not synchronized.

12) Difference between the Iterator and ListIterator

*Using Iterator we can traverse the list of objects only in forward direction . But ListIterator can traverse the collection in both directions that is forward as well as backward.
*ListIterator has .add() method whereas Iterator does not have.

13) Write a program to write employee object to the file and read it.

Employee Class Implements Serializable:

```
package properties;

import java.io.Serializable;


public class Employee implements Serializable {
	int EmpID;
	String EmpName;
	double Salary;
	int age;

	public Employee(int empID, String empName, double salary, int age) {
		super();
		EmpID = empID;
		EmpName = empName;
		Salary = salary;
		this.age = age;
	}

	@Override
	public String toString() {
		return "Employee [EmpID=" + EmpID + ", EmpName=" +
EmpName + ", Salary=" + Salary + ", age=" + age + "]";
	}

}
```

EmpObjectReadWrite Class:

```
package properties;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;

public class EmpObjectReadWrite  {
	public static void main(String[] args) throws IOException,
ClassNotFoundException {

		FileOutputStream fout=new FileOutputStream("object.txt");
		ObjectOutputStream objout = new ObjectOutputStream(fout);
```

```
                objout.writeObject(new Employee(10,"Ajay",30000,25));
                objout.writeObject(new Employee(20,"Bala",40000,27));
                objout.writeObject(new Employee(30,"Chandru",35000,26));
                objout.writeObject(new Employee(10,"Danush",50000,30));
                fout.close();

                FileInputStream fin=new FileInputStream("object.txt");
                ObjectInputStream objIn = new ObjectInputStream(fin);

                Employee emp =(Employee)objIn.readObject();
                System.out.println(emp);
                Employee emp1 =(Employee)objIn.readObject();
                System.out.println(emp1);
                Employee emp2 =(Employee)objIn.readObject();
                System.out.println(emp2);
                Employee emp3 =(Employee)objIn.readObject();
                System.out.println(emp3);
        }

}
```

14) Write a program to write employee array of objects to the file and read it.

Employee_Array Class Implements Serializable:

```
package properties;

import java.io.Serializable;


public class Employee_Array implements Serializable {
        int EmpID;
        String EmpName;
        double Salary;
        int age;

        public Employee_Array (int empID, String empName, double salary, int
age) {
                super();
                this.EmpID = empID;
                this.EmpName = empName;
                this.Salary = salary;
                this.age = age;
```

```
	}

	@Override
	public String toString() {
		return "Employee [EmpID=" + EmpID + ", EmpName=" +
EmpName + ", Salary=" + Salary + ", age=" + age + "]";
	}

}
```

EmpArrayObjReadWrite Class:

```
package properties;

import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.util.Scanner;

public class EmpArrayObjReadWrite {
	public static void main(String[] args) throws IOException,
ClassNotFoundException {

		Employee_Array [] empArray = new Employee_Array[3];
		FileOutputStream fout=new FileOutputStream("Arrayobject.txt");
		ObjectOutputStream objout = new ObjectOutputStream(fout);
		for(int i=0;i<empArray.length;i++)
		{
			int empID ;
			String empName ;
			double salary;
			int age ;
			Scanner sc =new Scanner(System.in);
			System.out.println("Enter the Empid EmpName Salary
Age");
			empID=sc.nextInt();
			empName=sc.next();
			salary=sc.nextDouble();
			age=sc.nextInt();
```

```
                    Employee_Array empArr=new
Employee_Array(empID,empName,salary,age);
                    empArray[i]=empArr;
                    objout.writeObject(empArray[i]);
                }
                objout.writeObject(new EOFile());
                fout.close();

                FileInputStream fin=new FileInputStream("Arrayobject.txt");
                ObjectInputStream objIn = new ObjectInputStream(fin);
                Object obj=null;
                while((obj=objIn.readObject()) instanceof EOFile==false)
                {
                    Employee_Array eA=(Employee_Array)obj;
                System.out.println(eA);
                }

        }
}

class EOFile implements Serializable
{

}
```