1) accept 10 numbers from user and add them inside the ArrayList.

using ListIterator display all the numbers bidirectionally.

import java.util.\*;

public class Demo {

public static void main(String[] args) throws Exception {

Scanner sc = new Scanner(System.***in***);

List <Integer> l = new ArrayList<>();

System.***out***.println("Enter 10 elements :");

for(int i=0;i<10;i++) {

l.add(sc.nextInt());

}

ListIterator<Integer> itr = l.listIterator();

while(itr.hasNext()) {

System.***out***.print(itr.next()+" ");

}

while(itr.hasPrevious()) {

System.***out***.print(itr.previous()+" ");

}

}

}

2) on the developer side:

create a generic class which can accept any type

with setters , getters and toString methods.

create necessary jar files and documentation.

on client side

create the object of above generic class and invoke its setters , getters and display the object.

package developer;

class GenericDemo <T>{

private T value;

public void setter(T value) {

this.value = value;

}

public T getter() {

return value;

}

public String toString() {

return "value : ["+value+"]";

}

}

package client;

import developer.GenericDemo;

public class Demo {

public static void main(String[] args) throws Exception {

GenericDemo<String> g= new GenericDemo<>();

g.setter("Hello");

System.***out***.println(g.getter());

System.***out***.println(g);

}

}

3) create LinkedList with the values 10,20,30 and 40.

display it.

now insert 500 in the beginning.

insert 400 at 2nd position.

add 1000 at the end.

display the list again.

import java.util.\*;

public class Demo {

public static void main(String[] args) throws Exception {

List<Integer> l = new LinkedList<>();

l.add(10);

l.add(20);

l.add(30);

l.add(40);

System.***out***.println(l);

((LinkedList) l).addFirst(500);

l.add(1,400);

((LinkedList) l).addLast(1000);

System.***out***.println(l);

}

}

4) add 5 numbers inside CopyOnWriteArrayList and show how can you add one more number at the same time of traversal through its iterator.

import java.util.\*;

import java.util.concurrent.\*;

public class Demo {

public static void main(String[] args) throws Exception {

CopyOnWriteArrayList<Integer> l = new CopyOnWriteArrayList<>();

l.add(10);

l.add(20);

l.add(30);

l.add(40);

l.add(50);

System.***out***.println(l);

Iterator<Integer> itr = l.listIterator();

while(itr.hasNext()) {

int num = itr.next();

System.***out***.println("now at : " + num);

if(num == 30) {

System.***out***.println("now adding element 60 ");

l.add(60);

}

}

System.***out***.println(l);

}

}

5) create a class MyNum with

private int num;

parameterized constructor

Demo class with main

create 4 objects of MyNum by passing different integers.

store these 4 objects inside "ArrayList"

and now store that ArrayList inside file system.

read ArrayList from file and traverse it using Iterator.

import java.io.\*;

import java.util.\*;

import java.util.concurrent.\*;

class MyNum implements Serializable{

private int num;

MyNum(int num){

this.num = num;

}

public String toString() {

return String.*valueOf*(num);

}

}

class Demo {

public static void main(String[] args) {

MyNum m1 = new MyNum(10);

MyNum m2 = new MyNum(20);

MyNum m3 = new MyNum(30);

MyNum m4 = new MyNum(40);

List<MyNum> l = new ArrayList<>();

l.add(m1);

l.add(m2);

l.add(m3);

l.add(m4);

String filepath = "D:\\cdac\\Java\\JavaSE\_Day\_10\\Assignments\\demo.txt";

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(filepath));){

oos.writeObject(l);

} catch (Exception e) { e.printStackTrace();}

List <MyNum> read=null;;

try(ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filepath));) {

read = (List<MyNum>) ois.readObject();

} catch (Exception e) { e.printStackTrace();}

Iterator<MyNum> itr = read.iterator();

while (itr.hasNext()) {

System.***out***.println(itr.next());

}

}

}

6)

Create a Student class with fields:

id (int)

name (String)

age (int)

marks (double)

Constructor & toString() method

Use an ArrayList<Student> to store student records.

Implement the following operations in the main function.

Add a student

Display all students

Search for a student by ID

import java.util.ArrayList;

import java.util.Scanner;

public class Student {

int id;

String name;

int age;

double marks;

Student(int id, String name, int age, double marks) {

this.id = id;

this.name = name;

this.age = age;

this.marks = marks;

}

public String toString() {

return "ID: " + id + ", Name: " + name + ", Age: " + age + ", Marks: " + marks;

}

public static void main(String[] args) {

ArrayList<Student> students = new ArrayList<>();

Scanner sc = new Scanner(System.***in***);

System.***out***.print("Enter ID: ");

int id = sc.nextInt();

sc.nextLine();

System.***out***.print("Enter Name: ");

String name = sc.nextLine();

System.***out***.print("Enter Age: ");

int age = sc.nextInt();

System.***out***.print("Enter Marks: ");

double marks = sc.nextDouble();

students.add(new Student(id, name, age, marks));

System.***out***.println("\nAll Students:");

for (Student s : students) {

System.***out***.println(s);

}

System.***out***.print("\nEnter ID to search: ");

int searchId = sc.nextInt();

boolean found = false;

for (Student s : students) {

if (s.id == searchId) {

System.***out***.println("Student Found: " + s);

found = true;

break;

}

}

if (!found) {

System.***out***.println("Student not found.");

}

}

}

7)

Create an interface Product with a method displayDetails().

Create two classes:

Electronics and Groceries

Both should implement Product.

Store objects of both the classes in a ArrayList<Product>.

store ArrayList in the file , then read ArrayList from the file and display.

import java.io.\*;

import java.util.ArrayList;

interface Product extends Serializable {

void displayDetails();

}

class Electronics implements Product {

String brand;

double price;

Electronics(String brand, double price) {

this.brand = brand;

this.price = price;

}

public void displayDetails() {

System.***out***.println("Electronics [Brand: " + brand + ", Price: " + price + "]");

}

}

class Groceries implements Product {

String name;

double weight;

Groceries(String name, double weight) {

this.name = name;

this.weight = weight;

}

public void displayDetails() {

System.***out***.println("Groceries [Name: " + name + ", Weight: " + weight + "kg]");

}

}

public class Main {

public static void main(String[] args) {

ArrayList<Product> products = new ArrayList<>();

products.add(new Electronics("Samsung TV", 45000));

products.add(new Electronics("Laptop", 60000));

products.add(new Groceries("Rice", 10));

products.add(new Groceries("Apples", 2));

String filepath = "D:\\cdac\\Java\\JavaSE\_Day\_10\\Assignments\\products.txt";

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(filepath))) {

oos.writeObject(products);

System.***out***.println("Products saved to file.");

} catch (Exception e) {

e.printStackTrace();

}

ArrayList<Product> readProducts = null;

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(filepath))) {

readProducts = (ArrayList<Product>) ois.readObject();

System.***out***.println("\nProducts read from file:");

for (Product p : readProducts) {

p.displayDetails();

}

} catch (Exception e) {

e.printStackTrace();

}

}

}

8) create a class "MyClass".

create 5 different objects of it.

add them in a ArrayList.

store ArrayList in file.

Now open a file, read ArrayList and display all objects.

import java.io.\*;

import java.util.ArrayList;

class MyClass implements Serializable {

int id;

String name;

MyClass(int id, String name) {

this.id = id;

this.name = name;

}

public String toString() {

return "MyClass [ID: " + id + ", Name: " + name + "]";

}

}

public class Main {

public static void main(String[] args) {

ArrayList<MyClass> list = new ArrayList<>();

list.add(new MyClass(1, "Object1"));

list.add(new MyClass(2, "Object2"));

list.add(new MyClass(3, "Object3"));

list.add(new MyClass(4, "Object4"));

list.add(new MyClass(5, "Object5"));

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("myclass.dat"))) {

oos.writeObject(list);

System.***out***.println("ArrayList stored in file.");

} catch (IOException e) {

e.printStackTrace();

}

ArrayList<MyClass> readList = null;

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream("myclass.dat"))) {

readList = (ArrayList<MyClass>) ois.readObject();

System.***out***.println("\nObjects read from file:");

for (MyClass obj : readList) {

System.***out***.println(obj);

}

} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

}

}

}

9) create a HashMap.

store prn no. and students name of 10 students inside the HashMap.

now write this HashMap inside the file and read also. After reading display it using iterator.

import java.io.\*;

import java.util.\*;

public class Main {

public static void main(String[] args) {

HashMap<Integer, String> students = new HashMap<>();

students.put(101, "Aman");

students.put(102, "Baban");

students.put(103, "Charan");

students.put(104, "Daman");

students.put(105, "Eran");

students.put(106, "Furan");

students.put(107, "Guran");

students.put(108, "Haran");

students.put(109, "Iran");

students.put(110, "Jaman");

try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream("students.dat"))) {

oos.writeObject(students);

System.***out***.println("HashMap stored in file.");

} catch (IOException e) {

e.printStackTrace();

}

HashMap<Integer, String> readStudents = null;

try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream("students.dat"))) {

readStudents = (HashMap<Integer, String>) ois.readObject();

System.***out***.println("\nHashMap read from file:");

Set<Map.Entry<Integer, String>> entrySet = readStudents.entrySet();

Iterator<Map.Entry<Integer, String>> iterator = entrySet.iterator();

while (iterator.hasNext()) {

Map.Entry<Integer, String> entry = iterator.next();

System.***out***.println("PRN: " + entry.getKey() + ", Name: " + entry.getValue());

}

} catch (IOException | ClassNotFoundException e) {

e.printStackTrace();

}

}

}

10) create a hierarchy as follows

interface Game- play() method

Derive at least 3 classes from it. ( Cricket, Chess and Football )

Now Create generic class which can take Game type (i.e. any sub type of Game) as parameter.

In this class try to call the play() method of the class which is passed to it.

interface Game {

void play();

}

class Cricket implements Game {

public void play() {

System.***out***.println("Playing Cricket...");

}

}

class Chess implements Game {

public void play() {

System.***out***.println("Playing Chess...");

}

}

class Football implements Game {

public void play() {

System.***out***.println("Playing Football...");

}

}

class GamePlayer<T extends Game> {

T game;

GamePlayer(T game) {

this.game = game;

}

void startGame() {

game.play();

}

}

public class Main {

public static void main(String[] args) {

GamePlayer<Cricket> cricketPlayer = new GamePlayer<>(new Cricket());

cricketPlayer.startGame();

GamePlayer<Chess> chessPlayer = new GamePlayer<>(new Chess());

chessPlayer.startGame();

GamePlayer<Football> footballPlayer = new GamePlayer<>(new Football());

footballPlayer.startGame();

}

}

11) maintain ICC ranking of at least 5 batsmen in the "HashMap".

list should be maintain as "rank" "name"

e.g. "1" "Rohit Sharma".

Now ask any rank bet'n 1 to 5 from user. ( user input).

if user asks for a particular rank , retrieve batsman name for that rank from the HashMap.

import java.util.HashMap;

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

HashMap<Integer, String> iccRanking = new HashMap<>();

iccRanking.put(1, "Rohit Sharma");

iccRanking.put(2, "Virat Kohli");

iccRanking.put(3, "Babar Azam");

iccRanking.put(4, "Kane Williamson");

iccRanking.put(5, "Steve Smith");

Scanner sc = new Scanner(System.***in***);

System.***out***.print("Enter rank (1-5): ");

int rank = sc.nextInt();

if (iccRanking.containsKey(rank)) {

System.***out***.println("Rank " + rank + " -> " + iccRanking.get(rank));

} else {

System.***out***.println("Invalid rank! Please enter between 1 and 5.");

}

sc.close();

}

}

12)

Create a base class Animal with makeSound().

Create subclasses Dog and Cat, each overriding makeSound().

Store objects of Dog and Cat in an ArrayList<Animal>.

Now traverse the ArrayList using iterator and call makeSound() dynamically.

import java.util.ArrayList;

import java.util.Iterator;

class Animal {

void makeSound() {

System.***out***.println("Animal makes a sound");

}

}

class Dog extends Animal {

void makeSound() {

System.***out***.println("Dog barks");

}

}

class Cat extends Animal {

void makeSound() {

System.***out***.println("Cat meows");

}

}

public class Main {

public static void main(String[] args) {

ArrayList<Animal> animals = new ArrayList<>();

animals.add(new Dog());

animals.add(new Cat());

Iterator<Animal> it = animals.iterator();

while (it.hasNext()) {

Animal a = it.next();

a.makeSound();

}

}

}