a) Memento design pattern

Here previous states of the contestants could be able to restore.

There should have a class to keep track of the states of the contestants.

Memento pattern is applicable for this scenario.

b) Flyweight design pattern

As contestants are complex objects and their some attributes are same and

Not change throughout the game, it is memory inefficient to create those complex objects.

Instead they can share those attributes and only keep changeable attributes. In this

Design pattern one complex object is created and it is shared by all others.

==================================================================================

import java.util.ArrayList;

import java.util.HashMap;

public class GameManager {

private HashMap<Integer,ArrayList<Memento>> mementolist;

public GameManager(){

mementolist=new HashMap<Integer,ArrayList<Memento>>();

}

public void addMemento(int ID,Memento memento){

mementolist.get(ID).add(memento);

}

public void addMementoList(int ID) {

mementolist.put(ID,new ArrayList<Memento>());

}

public Memento setPreviousState(int ID) {

mementolist.get(ID).remove(mementolist.get(ID).size()-1);

return mementolist.get(ID).get(mementolist.get(ID).size()-1);

}

}

public class Memento {

private int PositionX,PositionY;

public Memento(int X,int Y){

PositionX=X;

PositionY=Y;

}

public int getX() {

return PositionX;

}

public int getY() {

return PositionY;

}

}

public class FlyweightFactory {

private static FlyweightContestant contestant;

public static FlyweightContestant createContestant(){

if (contestant==null){

synchronized(FlyweightFactory.class){

if (contestant==null){

contestant= new FlyweightContestant();

}

}

}

return contestant;

}

}

public class FlyweightContestant {

private String Colour="Grean shirt and Black trouser";

private GameManager manager;

}

public class Contestant {

private int PositionX,PositionY;

private int ID;

private Memento memento;

private FlyweightContestant contestant;

private GameManager manager;

public Contestant(int ID,GameManager manager){

this.ID=ID;

contestant=FlyweightFactory.createContestant();

this.manager=manager;

manager.addMementoList(ID);

}

public void saveState(){

manager.addMemento(ID,new Memento(PositionX,PositionY));

}

public void setPreviousState(){

memento=manager.setPreviousState(ID);

PositionX=memento.getX();

PositionY=memento.getY();

}

public int getX(){

return PositionX;

}

public int getY(){

return PositionY;

}

public void setX(int x){

PositionX=x;

}

public void setY(int y){

PositionY=y;

}

}

package r\_2017question1;

public class R\_2017Question1 {

public static void main(String[] args) {

GameManager manager=new GameManager();

Contestant contestant1=new Contestant(1,manager);

contestant1.setX(10);

contestant1.setY(10);

contestant1.saveState();

Contestant contestant2=new Contestant(2,manager);

contestant2.setX(15);

contestant2.setY(15);

contestant2.saveState();

Contestant contestant3=new Contestant(3,manager);

contestant3.setX(20);

contestant3.setY(20);

contestant3.saveState();

contestant1.setX(30);

contestant1.setY(40);

contestant1.saveState();

contestant1.setX(25);

contestant1.setY(35);

contestant1.saveState();

System.out.println("x :"+contestant1.getX()+" y :"+contestant1.getY());

contestant1.setPreviousState();

System.out.println("x :"+contestant1.getX()+" y :"+contestant1.getY());

contestant1.setPreviousState();

System.out.println("x :"+contestant1.getX()+" y :"+contestant1.getY());

}

}