*// A Java program to demonstrate working of*

*// FlyWeight Pattern with example of Counter*

*// Strike Game*

import java.util.Random;

import java.util.HashMap;

*// A common interface for all players*

interface Player

{

    public void assignWeapon(String weapon);

    public void mission();

}

*// Terrorist must have weapon and mission*

class Terrorist implements Player

{

*// Intrinsic Attribute*

    private final String TASK;

*// Extrinsic Attribute*

    private String weapon;

    public Terrorist()

    {

        TASK = "PLANT A BOMB";

    }

    public void assignWeapon(String weapon)

    {

*// Assign a weapon*

        this.weapon = weapon;

    }

    public void mission()

    {

*//Work on the Mission*

        System.out.println("Terrorist with weapon "

                           + weapon + "|" + " Task is " + TASK);

    }

}

*// CounterTerrorist must have weapon and mission*

class CounterTerrorist implements Player

{

*// Intrinsic Attribute*

    private final String TASK;

*// Extrinsic Attribute*

    private String weapon;

    public CounterTerrorist()

    {

        TASK = "DIFFUSE BOMB";

    }

    public void assignWeapon(String weapon)

    {

        this.weapon = weapon;

    }

    public void mission()

    {

        System.out.println("Counter Terrorist with weapon "

                           + weapon + "|" + " Task is " + TASK);

    }

}

*// Class used to get a player using HashMap (Returns*

*// an existing player if a player of given type exists.*

*// Else creates a new player and returns it*.

class PlayerFactory

{

*/\* HashMap stores the reference to the object*

*of Terrorist(TS) or CounterTerrorist(CT).  \*/*

    private static HashMap <String, Player> hm =

                         new HashMap<String, Player>();

*// Method to get a player*

    public static Player getPlayer(String type)

    {

        Player p = null;

*/\* If an object for TS or CT has already been*

*created simply return its reference \*/*

        if (hm.containsKey(type))

                p = hm.get(type);

        else

        {

*/\* create an object of TS/CT  \*/*

            switch(type)

            {

            case "Terrorist":

                System.out.println("Terrorist Created");

                p = new Terrorist();

                break;

            case "CounterTerrorist":

                System.out.println("Counter Terrorist Created");

                p = new CounterTerrorist();

                break;

            default :

                System.out.println("Unreachable code!");

            }

*// Once created insert it into the HashMap*

            hm.put(type, p);

        }

        return p;

    }

}

*// Driver class*

public class CounterStrike

{

*// All player types and weapon (used by* *getRandPlayerType()*

*// and getRandWeapon()*

    private static String[] playerType =

                    {"Terrorist", "CounterTerrorist"};

    private static String[] weapons =

      {"AK-47", "Maverick", "Gut Knife", "Desert Eagle"};

*//* *Driver* *code*

    public static void main(String args[])

    {

*/\* Assume that we have a total of 10 players*

*in the game. \*/*

        for (int i = 0; i < 10; i++)

        {

*/\* getPlayer() is called simply using the class*

*name since the method is a static one \*/*

            Player p = PlayerFactory.getPlayer(getRandPlayerType());

*/\* Assign a weapon chosen randomly uniformly*

*from the weapon array  \*/*

            p.assignWeapon(getRandWeapon());

*// Send this player on a mission*

            p.mission();

        }

    }

*// Utility methods to get a random player type and*

*// weapon*

    public static String getRandPlayerType()

    {

        Random r = new Random();

*// Will return an integer between [0,2*)

        int randInt = r.nextInt(playerType.length);

*// return the player stored at index 'randInt'*

        return playerType[randInt];

    }

    public static String getRandWeapon()

    {

        Random r = new Random();

*// Will return an integer between [0,5)*

        int randInt = r.nextInt(weapons.length);

*// Return the weapon stored at index 'randInt'*

        return weapons[randInt];

    }

}

Counter Terrorist Created

Counter Terrorist with weapon Gut Knife| Task is DIFFUSE BOMB

Counter Terrorist with weapon Desert Eagle| Task is DIFFUSE BOMB

Terrorist Created

Terrorist with weapon AK-47| Task is PLANT A BOMB

Terrorist with weapon Gut Knife| Task is PLANT A BOMB

Terrorist with weapon Gut Knife| Task is PLANT A BOMB

Terrorist with weapon Desert Eagle| Task is PLANT A BOMB

Terrorist with weapon AK-47| Task is PLANT A BOMB

Counter Terrorist with weapon Desert Eagle| Task is DIFFUSE BOMB

Counter Terrorist with weapon Gut Knife| Task is DIFFUSE BOMB

Counter Terrorist with weapon Desert Eagle| Task is DIFFUSE BOMB