ADVANCED JAVA PROGRAMMING AJP-P2-2012-2013

INTRODUCTION

In this exercise your knowledge of interfaces, abstract classes and the static modifier will be tested. You will be given a set of unit tests. Your job is to write the code that satisfies the tests. You will also learn about the *enum* type.

SET-UP

Import the Netbeans project folder named 'AJP-P2-2012-2013-STUDENT'. Open it up. You have been given a set of unit tests. Initially, there will be LOTS of syntax errors in these unit tests. This is expected. The errors will disappear as you write the required classes.

INSTRUCTIONS FOR BRONZE AWARD

You are working on a video game.

In this game there are three types of enemy soldier – cadets, majors and colonels.

There are also two types of vehicle – the light tank and the heavy tank.

There are also three factions (armies) within the game – NOD, GDI and SCRIN

When users play the game they try to kill soldiers and destroy vehicles.

When a player kills a soldier, or destroys a vehicle, he/she scores points, as shown below:

	GDI	NOD	SCRIN
CADET	25	20	15
MAJOR	55	60	70
COLONEL	100	95	90
LIGHT TANK	150	175	150
HEAVY TANK	225	310	270

Furthermore, when a soldier is killed, or a vehicle is destroyed, a text message is generated to alert the user e.g.

A SCRIN heavy tank is demolished: +270pts.

Now you will develop a solution that satisfies all of the unit tests in the code project.

Your solution must have one interface, two abstract classes, 5 concrete classes and one *enum* type.

First of all, write an *enum* called *Faction* that defines the three factions in the game. You can learn more about *enums* on p.331 of Deitel and Deitel (see e-library).

Now write the interface *Killable* with a single method *kill*(). The return type for this method should be *String*.

Now write the abstract class called *Soldier*. This abstract class should implement the *Killable* interface and define a *getPoints*() method with an *int* return type. The constructor for this class should take an *enum* constant of type *Faction*.

Now extend the abstract class *Soldier* in three concrete classes, *Cadet*, *Major* and *Colonel*. Examine the unit tests closely to discover how these classes should be written. For example, *CadetTest.java* tells us that the *kill*() method of the *Cadet* class should return

```
A NOD cadet bites the dust: +20pts.\n
```

If the cadet is part of the NOD faction.

Now create the abstract class called *Vehicle*. This abstract class should implement the *Killable* interface and define a *getPoints*() method with an *int* return type. The constructor for this class should take an *enum* constant of type *Faction.*

Extend the abstract class *Vehicle* in two concrete classes, *LightTank* and *HeavyTank*. Examine the unit tests closely to discover how these classes should be written.

The constructors for all concrete classes should call the super-constructor (i.e they should all call either *Vehicle* or *Soldier*), passing an *enum* constant of type *Faction*.

You will also need to create two counters that track the total number of points scored for killing soldiers and destroying vehicles. These counters should be declared in the abstract classes.

When all unit tests pass, you have a BRONZE award.

INSTRUCTIONS FOR SILVER AWARD

If you would like to try for a SILVER award, make sure that your *CheckStyle* report has no errors.

INSTRUCTIONS FOR GOLD AWARD

Create a GUI that tests your new class library.

Your GUI should have a large red button at its centre.

When this button is pressed, you will generate 1-5 random *Soldier* and *Vehicle* objects i.e. random classes, random factions.

The number and type of kills should be illustrated graphically using suitable images (download images or create your own).

Keep track of the number of points scored each time the button is pressed.

Keep track of the highest score ever achieved.

If the number of points is larger than the previous high score, update the high score.

Here is a mock up of the sort of application you need to create.