Software and Programming I

Lab 8:
Use of classes, static class variables and methods

Tobi Brodie (Tobi@dcs.bbk.ac.uk)

Lab 8 Objectives

Understanding the encapsulation of data in objects.

Use of the debugger to follow changes in objects.

Understanding the difference between static (belongs to the class) and non-static (belongs to a particular object) variables and methods.

Scan program descriptions and break them down into the correct code.

Note: Classes **Team / Game / GameTest** are a single Marked Exercise 6 and you are required to complete them, and show the finished program, explaining how they work together, by the 19th of March. *Failure to complete ALL exercises by this date will mean you will not receive the full coursework marks.*

Exercise 1: T09

To complete this exercise you will need to download the code from the following URL:

http://www.dcs.bbk.ac.uk/~roman/sp1/extra/T09.java

Once the code is downloaded, right click on the file and choose *edit with notepad++* from the contextual menu.

Copy this code into a newly created class T09 in BlueJ.

Do not run the code!

Exercise 1: T09 (2)

DO NOT RUN THE CODE!

First, analyse the code by hand and write down below what you think will be printed out:

Compile the code, and then choose breakpoints that you can use with the BlueJ debugger to see when values change within the objects/class.

Now, run the `main` method and check the results.

Marked Exercise 6: Team

Implement class `Team` that represents a football team.

http://www.dcs.bbk.ac.uk/~roman/sp1/java/Team.java

A team has a name, a number of games played, a number of points and a total goal difference (which can be used to rank teams if they have the same number of points). These instance variables are already declared in the class.

The **constructor** of the class `Team` requires its name; other instance variables should get appropriate values.

You also need to declare all 4 **getters** for the 4 instance variables.

Marked Exercise 6: Team (2)

Finally, you will need to implement **method**

```
void addResult(int goalsFor, int goalsAgainst)
```

which is invoked every time a game is finished. Your implementation should

- update the number of games and the goal difference;
- increase the number of points by 3 for a win (when the team scores more goals than concedes);
- increase the number of points by 1 for a draw (same number of goals scored and conceded).

Marked Exercise 6: Game

Implement class `Game` that represents a football game between two teams

http://www.dcs.bbk.ac.uk/~roman/sp1/java/Game.java

The template contains method headers for the 5 **getters** (`getDate`, `getTeamH`, `getTeamG`, `getGoalsH`, `getGoalsG`). You will have to declare appropriate instance variables and implement the 5 getters.

Marked Exercise 6: Game (2)

The **constructor** requires a date and two teams. These parameters should be stored in the instance variables. In addition, the newly constructed instance of `Game` needs to be added to the array `allGames`, which is a class variable: the array should contain all games. You may need to declare additional class variables to ensure that the `Game` object is stored at an appropriate index.

You will also have to implement **class methods** `getGames()` and `getGames(String date)`: the second method should return a *new* array that contains all games on the specified date.

Marked Exercise 6: Game (3)

Implement **instance methods** `goalH()` and `goalG()` that will be invoked every time the respective team scores a goal. The implementations should adjust appropriate instance variables.

Finally, implement **instance method** `whistle()` that will be invoked when the game is finished. The implementation should use methods of class `Team` to update their statistics.

Marked Exercise 6: GameTest

Class `GameTest` contains code to create a few instance of classes `Team` and `Game` and invoke their methods

http://www.dcs.bbk.ac.uk/~roman/sp1/java/GameTest.java

This class should **not** be modified, but its code is worth studying.

Your implementations should be such that `GameTest` compiles (with your `Team` and `Game`), runs, and all tests in `GameTest` print `"PASS"`.

Home Work
Java for Everyone by C. Horstmann

Read Sections 9.1–9.4, which are available online from

http://vufind.lib.bbk.ac.uk/vufind/Record/566484

and complete the following exercises:

- Exercise R9.6
- Exercise R9.8
- Exercise P9.2
- Exercise P9.4