CMPSCI 187: Programming With Data Structures

Lecture #9: The Trivia Game Application
David Mix Barrington
24 September 2012

The Trivia Game Application

- Designs With Multiple Classes
- Sources of Classes
- The Overall Trivia Game Project
- Choosing Classes: TriviaGame and TriviaQuestion
- Implementing TriviaGame and TriviaQuestion
- Loading the Game From a File
- Playing the Game with a User

Designs With Multiple Classes

- A simple Java program usually consists of a single class. But substantial programs can take advantage of the object-orientation of Java to organize the code into multiple classes.

- When faced with a computational task to solve with a Java program, our first task is to decide *what classes* we need to define. What data will the program manipulate, and how should this data be organized?

- A good way to begin is to **brainstorm**, writing down possible classes to use, and then **filter**, identifying the most promising choices.

- DJW also suggest writing an English description of what the program is to do. The **nouns** in this text are good prospects for being classes, and the **verbs** are good prospects for being methods.

Sources of Classes

- Where do we find our classes?

- The **Java class library** contains versions of all the data structures we will use in the course, and many others. They have been written and tested by professionals, and carefully documented in the Java API. Of course, to be able to pick a useful class out, you have to have an idea of what is there and an idea of what you want. In 187 we are a long way from that kind of familiarity. But DJW list a useful *subset* of the library in Appendix E.

- If you have ever solved a similar problem, you may have already built a class that you can reuse or adapt. This is more likely to happen if you kept to the principal that classes should perform general jobs, and do them well.

- You may be able to find an **off the shelf** package with classes you can use. Copyright law comes in here -- to use someone else's work legally you may have to pay. But a lot of nice software is in the **public domain**.

The Overall Trivia Game Project

- Suppose you're planning a party and you plan to amuse your guests by having them play a trivia game with questions about people in reality-TV shows. You could write a program to do this, but the subject of the questions will not likely be a factor in your programming.

- You write down your task: "The game will present the player with a series of questions. For each question it will say whether the answer is among the possible correct answers. It will keep score of how many the player gets right, and stop after a given number of questions."

- We have nouns: "game", "player", "question", "answer", "correct answer", "number right", etc., and verbs like "present", "say whether", "keep score", and "stop".

- Which of these give us good candidates for classes?

Choosing Classes: TriviaGame and TriviaQuestion

- We have an entity called a “game” -- if we ask questions about soccer players at the next party, we might be using the same software to play a different “game”.

- Let’s make a TriviaGame class, where a TriviaGame object will have a name, some possible topics, a list of questions, and a set of valid answers for each question. The game also needs parameters like the number of questions to be asked, the number asked already (to tell when to stop), and the player’s score so far.

- It probably makes sense to make each question an object, with its topic, text, and set of answers. So we would appear to need a TriviaQuestion class as well.

- There are more details like the input and output, but the essence of the game is the questions and answers, and the keeping of the score.

Implementing TriviaGame and TriviaQuestion

- DJW's code for their TriviaGame class is on pages 136-8. They have int fields for the current and maximum number of questions, the number of chances, the remaining chances, the number correct, the number incorrect. (They allow multiple tries at a question if the first try is incorrect.) They have an array of TriviaQuestion objects and an array of booleans to record the result for each question. (Here `maxNumQuestions` is the size of the array, while `currNumQuestions` is the number of questions loaded into it.)

- A TriviaQuestion object has a category and a question text (both String objects), and the set of correct answers. What do we need from the set of answers? We want to add new correct answers, and test whether the player's answer is in the set. This is a perfect situation for a StringLog object.

- Neither of these classes have main methods. The action will start somewhere else, and TriviaGame and TriviaQuestion objects will be created and used.

Loading Questions From a File

- How do we get the questions into the game? There's no reason why your local expert on reality-TV shows would know anything about Java. You want your questions and answers from them in text form, ideally in a text **file** that your program can read. You can specify the format yourself, and have your expert follow it or edit what they give you.

- We could make a constructor for TriviaGame that took a file as a parameter and loaded the questions. But DJW choose to make a new class called GetTriviaGame, with a method that takes a file and returns a TriviaGame. This is because constructing the TriviaGame and loading it are separate jobs.

- File I/O in Java opens up various cans of worms, only some of which are kept under control by using the Scanner class. For example, our method that reads the text file must have a `throws IOException` clause because it uses the FileReader constructor and this exception is **checked**.

Playing the Game With a User

- We still haven't involved the player of the game. What they want is to interact with a program that uses the data loaded into the TriviaGame object to ask them questions and evaluate the answers. This is not the job of the TriviaGame object itself -- normally we separate the interactive parts of a program into a separate class, so that the fundamental classes focus on their basic operations that are independent of how the user interaction works.

- We have a TriviaConsole class that is easy to implement because it can call methods of the other classes for nearly all its activities. It has a main method. It expects the game data to be in a file called `game.txt` and loads it into a new TriviaGame object using the method `GetTriviaGame.useTextFile`. It plays the game by calling methods of the TriviaGame class and communicating with the player through the system input and output.

- If we wanted a GUI rather than a command-line version of such a game, we could reuse almost all our code, only replacing TriviaConsole with TriviaGUI.