Project 9

In this project you will write a program in *Jack*: A simple, Java-like, object-based programming language. This will set the stage for subsequent stages in our journey, in which we will build a Jack compiler, and a basic operating system.

Objective

The "hidden agenda" of this project is to get acquainted with the Jack language, for two subsequent landmarks: writing a Jack compiler in projects 10 and 11, and developing a basic operating system in project 12 (the OS will be written in Jack, just like modern OSs are written in high-level languages like C++). In addition, you'll become familiar with the art of writing a program that combines graphics, animation, and user interaction – a useful skill in and of itself.

Contract

Invent or adopt a simple computer game or some other interactive program, and implement it in the Jack language. Examples include basic versions of Tetris, Snake, Space Invaders, Sokoban, Pong, or simpler games like Hangman. More examples (some of them quite ambitious and impressive) can be viewed in the "Cool Stuff" section of the Nand2Tetris website. Note that you don't have to create a complete application. For example, you can create a basic version, or part of, some simple game or cool interaction.

Compiling and Running a Jack Program

0. Create a folder for your program. Let's call it the *program folder*.
1. Write your Jack program – a set of one or more Jack classes – each stored in a separate className.jack text file. Keep all these .jack files in the same program folder.
2. Compile the program folder using the supplied Jack compiler. This will cause the compiler to translate all the .jack classes found in the folder into corresponding .vm files, stored in the same folder. If a compilation error is reported, debug the program and re-compile until no error messages are issued.
3. At this point the program folder should contain your source .jack files, along with the compiled .vm files. To test the compiled program, load the program folder into the supplied VM emulator. Then run the program in the VM emulator. If you encounter run-time errors or undesired program behavior, fix the program and go to stage 2.

The Jack OS

Writing Jack programs requires working with the Jack OS, just like writing Java programs requires working with the Java class library. The Jack OS is a set of libraries that extend the basic language's capabilities and close gaps between it and the underlying hardware. Here is the Jack OS API, and the list of OS error codes and their meaning.

We supply two Jack OS implementations: "native", and "builtin". The native OS implementation was written in Jack and was then translated (using a Jack compiler) into the set of eight VM files

stored in the nand2tetris/tools/os folder in your PC. The builtin OS implementation was written in Java, and is embedded in the VM emulator available in nand2tetris/tools.

Which OS version to use is up to you. The builtin version is faster. The VM emulator does not care which OS version is used, for the following reason: Suppose you've loaded a program folder into the VM emulator, and proceeded to execute it. Whenever the emulator detects a call to some OSclass.function (e.g. Math.sqrt), it checks if this function is part of the loaded code base; if so, it executes this function's VM code; otherwise, it reverts to using the built-in implementation of this OS function.

Resources

You will need the supplied tools/JackCompiler, for translating your program into a set of .vm files, and the supplied tools/VMEmulator, for running and testing the compiled code.

Bitmap editor: If you develop a program that needs high-speed graphics, it is recommended to design *sprites* – repeating graphical images – for fast rendering of the key graphical elements of your program. Such sprites, along with the Jack code for rendering them, can be designed and generated, respectively, using this Bitmap Editor, created by Eric Umble. Here is a demo of a Dino Adventure game that Eric developed with this editor (a simpler desktop bitmap editor, developed by Golan Parashi, is also available in the nand2tetris/tools folder).

In addition to these resources, you will need resourcefulness – coming up with a compelling idea for an interactive application or a simple game. This Youtube channel contains examples of "retro" computer games. You can review it for ideas, as well as for technical tips.

Code example: The projects/09/Square folder includes the source code of a complete, 3-class interactive Jack program. This program illustrates various programming techniques which are commonly used in designing applications that combine graphics, animation, and user interaction. Therefore, it is recommended to read and play with this program before starting to work on your own program.

Submission and grading

Submit a folder containing the source .jack files of your application. We will compile your files and judge your work along the following criteria: Code quality and documentation (40%), user experience (50%), and originality (10%). Real cool programs will get an additional 10% bonus.