CS4120/4121/5120/5121—Spring 2016
Programming Assignment 7
Object-Oriented Features

Due: Thursday, May 19, 11:30AM

HARD DEADLINE, NOT MIDNIGHT

This assignment requires to you add object-oriented features, completing your compiler. The new, extended language is called oXi, which adds classes and inheritance to Xi. The differences between oXi and Xi are described in oXi Language Specification. This assignment also requires you to extend oXi with a new feature of your choosing.

Unlike with the previous assignments, the due date for this assignment is a hard deadline. Following university rules, the assignment is also not due at midnight.

0 Changes

- 5/17: Updated `xth`.
- 5/9: The extended compiler will be run from a new command line interface `oxic`. The old command line interface `xic` could either recognize the extended features or support only pre-extension features.

1 Instructions

1.1 Grading

Solutions will be graded on documentation and design, completeness of the implementation, correctness, and style. 10% of the score is allocated to whether bugs in past assignments have been fixed.

1.2 Partners

You will work in a group of 3–4 students for this assignment. This should be the same group as in the last assignment. If not, please discuss with the course staff.

Remember that the course staff is happy to help with problems you run into. For help, read all Piazza posts and ask questions (that have not already been addressed), attend office hours, or meet with any course staff member either at the prearranged office hour time or at a mutually satisfactory time you arrange.

1.3 Package names

Please ensure that all Java code you submit is contained within a package whose name contains the NetID of at least one of your group members. Subpackages under this package are allowed; they can be named however you would like.

2 Design overview document

We expect your group to submit an overview document. The Overview Document Specification outlines our expectations.

3 Building on previous programming assignments

As before, you are building upon your work from Programming Assignment 6. The protocol is the same as in prior assignments: you are required to develop and implement tests that expose any problems with your implementation, and then fix the problems. Correctness of previous assignments will count for more than it has earlier.

4 Version control

As in the last assignment, you must submit file `pa7.log` that lists the commit history from your group since your last submission.

5 Updates to provided code

An updated version of the `libxi` runtime library is needed for development with oXi. The `xifilt` utility has been updated and now understands oXi symbols.

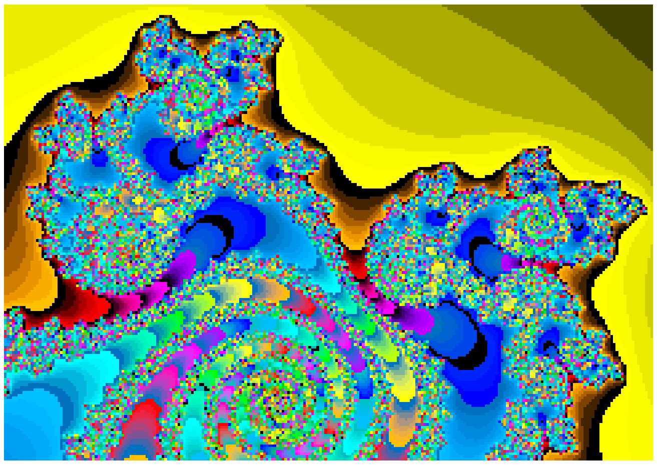
We are also providing two object-oriented example programs and corresponding assembly code. These code resources will be made available soon.

The QtXi GUI Library enables oXi programs to build graphical user interfaces that work cross-platform, such as the Mandelbrot example below.

6 Example programs

Some example programs are available for testing your compiler:

- mandelbrot: A graphical Mandelbrot set explorer using QtXi.



- kmp: A Knuth-Morris-Pratt string matcher (mostly not OO, but uses the new file interface).

7 Language extension

We expect you to design and implement some new, small feature for oXi. Some examples of features you might add are the following, roughly in order of difficulty.

- Allow constants to be defined in interface and source files, integrated with constant folding (relatively easy).
- Support more powerful multiple-assignment syntax, with multiple expressions appearing on the right-hand side (relatively easy).
- Java-style `break` and `continue` that can jump to named labels.
- Make Xi type-safe by detecting uninitialized variables.
- Multiple inheritance with sparse dispatch vectors (relatively challenging).

But you don't have to implement one of these choices—feel free to be creative! It doesn't have to be a big feature. You will get a bit more credit for attempting more complex features, but you will also get credit for building a rock-solid implementation of your feature and for documenting it clearly. So don't bite off more than necessary.

Your language should be backward-compatible with the oXi spec so correct programs written according to that spec still work.

8 Command-line interface

You must create a new command line interface called `oxic` that compiles oXi programs. The existing command line interface `xic` could recognize either oXi programs or only programs that uses pre-extension features.

The command-line syntax is as follows:

```
oxic [options] <source files>
```

The possible options are as defined in the previous assignment.

9 Build script

Your build script `xic-build` from previous programming assignments should remain available. The expected behaviors of the build script are as defined in the previous assignment. **The build script must be in the root directory of your submission `zip` file.** Problems within the build script from previous submissions should be fixed.

10 Test harness

`xth` has been updated to contain test cases for this assignment and to support testing language extension.

To update `xth`, run the `update` script in the `xth` directory on the VM.

A general form for the `xth` command-line invocation is as follows:

```
xth [options] <test-script>
```

The following options are of particular interest:

- `-compilerpath <path>`: Specify where to find the compiler
- `-testpath <path>`: Specify where to find the test files
- `-workpath <path>`: Specify the working directory for the compiler

For the full list of currently available options, invoke `xth`.

The best way to run `xth` with the provided test cases is from the home directory of the VM, using the following form of command:

```
xth -compilerpath <xicpath> -testpath <tp> -workpath <wp> <xthScript>
```

where

- `<xicpath>` is the path to the directory containing your build script and command-line interface.
- `<tp>` is of the form `xth/tests/pa#/`, where # is the programming assignment number.
- `<wp>` is preferably a fresh, nonexistent compiler such as `shared/xthout`.
- `<xthScript>` is of the form `xth/tests/pa#/xthScript`, where # is the programming assignment number.

An `xth` test script specifies a number of test cases to run. Once the updated `xth` is released, directory `xth/tests/pa7` will contain a sample test script (`xthScript`), along with several test cases. `xthScript` also lists the syntax of an `xth` test script.

Despite additional testing, `xth` is still in the development phase. Many features have been added since the last release. Please report errors, request additional features, or give feedback on Piazza.

11 Functional and benchmark test cases

The compilers from various groups will be compared in the 2016 CS 4121/5121 Compiler Bakeoff. The winning compiler time will gain bragging rights and be immortalized on the web page above.

For fun and good karma, you are encouraged to submit programs that you think are good functional tests or good performance benchmarks. We encourage each group to submit up to five functional test cases and up to three benchmark test cases.

Each test case must be a valid oXi source file. A compiler `c` passes a test `t` if and only if

- `c` successfully compiles `t` into an assembly file `a`,
- assembling and linking `a` against the standard Xi library results in a runnable program `o`, and
- when executed, `o` terminates with exit code 0 **within 3 seconds**; i.e., it terminates normally, and not as a result of an assertion failing or an array-out-of-bounds violation.

All test cases must

- be ASCII-encoded files,
- be valid oXi programs, according to the standard specifications (i.e., the oXi language specification),
- not read input,
- contain at most 20 lines of code, excluding comments, and
- contain no line longer than 80 characters.

These test cases will also be run against the compilers of other groups, and groups will receive good karma for generating the fastest code for submitted test cases, or for submitting test cases that expose bugs in other compilers.

12 Submission

You should submit these items on CMS:

- `overview.txt/pdf`: Your overview document for the assignment. This file should contain your names, your NetIDs, all known issues you have with your implementation, and the names of anyone you have discussed the homework with. It should also include descriptions of any extensions you implemented.
- A `zip` file containing these items:
  - *Source code*: You should include all source code required to compile and run the project. Please ensure that the directory structure of your source files is maintained within the archive so that your code can be compiled upon extraction. If your code depends on any third-party libraries, please include compilation instructions in your overview document.
    Include your parser generator input file, e.g., `*.cup`, as well as the generated code. If you use a lexer generator, please include the lexer input file, e.g., `*.flex`, as well as the generated code.
  - *Tests*: You should include all your test cases and test code that you used to test your program. Be sure to mention where these files are and to describe your testing strategy in your overview document.
  - *Functional and benchmark test cases*: **These test cases must reside in directory `oxicontest` at the root of the `zip` file directory hierarchy.**

Do not include any non-source files or directories such as `.class`, `.classpath`, `.project`,

`.git`, and `.gitignore`.

- `pa7.log`: A dump of your commit log since your last submission from the version control system of your choice.