CSC 2/453 Dynamic Languages and Software Development

Chen Ding

Department of Computer Science
University of Rochester

(draft document in preparation)

Chapter 1

Introduction

1.1 Pre-lecture Preparation

1. Consider your answer or someone's answer that you find by a web search for the following questions

   - What is design?
   - What makes one design better than another?
   - How is designing software different from designing hardware, buildings, graphical user interface, etc?

2. Of the software you have used,

   - What is your favorite example?
   - What are the necessarily qualities for its success?
   - Do you have a favorite inventor/designer or a favorite quote?

3. Try downloading the class repository following the instructions in this chapter. Bring to class any problems and questions.

1.2 What is Design

The content will be added after the lecture.

1.3 Why 2/453

The content will be added after the lecture.

1.4 Characteristics of a Great Designer

The content will be added after the lecture.

1.5 Testimonials

These are some of the comments my past students made through the anonymous course evaluation about my style of teaching in related courses. Since this document is a written record, I include only the positive comments. Be aware that for almost every course you can find a student who likes it. Still it is useful to see a few views from the other side of the lecturer table/lectern.

- "[the lectures] make student think in class,"
- "The Prof. is really good at teaching the essence of ideas interactively without getting into boring details. I learned a lot from the class,"
- "In-class constructive examples are very very helpful. Open, collaborative atmosphere."
- "I was able to participate very well in the class. The lecture part of where the professor is speaking and the interactive part where we all collaborate in class to write a class worked out very well. The balance between the two was very well that I never spaced out."
- "This is the first time I've enjoyed a systems course. It was engaging, challenging, and attending the lectures was actually nontrivially useful"
- "Working on a large project, planning and working together, was a useful experience. I appreciated the freedom we had in the work."

1.6 Grading

- midterm and final exams are 15% each
- the projects total to 60%
- written assignments are 10%

Ruby will be the language used in class instruction and demonstration. However for all projects students are free to use any programming language of their choice, including non-dynamic languages C/C++/Java if they choose.

Graphical user interface (GUI) is nice but can take a significant amount of time to implement. This is not the emphasis of the course. It is preferable if the submitted projects have well-defined text-based interface. An assignment may specify such interface for automatic testing/grading.

1.7 Class Repository

The class repository is managed by the distributed version control system called Mercurial. It is distributed because your local machine has a copy of the repository. You check out files from the local repository and check in changes into it, all on the local machine without connecting to a network. To communicate between repositories on different machines, you use *hg push* and *hg pull* (*hg* is the chemical symbol of mercurial).

If you have not used Mercurial before, it is easy to follow one of the many on-line tutorials. The basic commands you need are as follows.

Before you start, make sure that you have an account on *cycle1.csug.rochester.edu*. Create your local repository in a directory, which we use *cs253hg* as an example. It is easiest to use a command-line interface through either the terminal program on Linux, Mac OS, or Cygwin on Windows. The sequence of commands are

```
mkdir cs253hg
cd cs253hg
hg init
```

Next link the new repository with the shared class repository on *cycle1.csug*. Create the file *cs253hg/.hg/hgrc* and add the following (if your repository is not on the same network as cycle1, e.g. on your laptop). Otherwise use *file://u/cding/cs253hg* instead of the ssh address below. If ssh is used, you will need to type in your password each time you synchronize with server repository (your local commits never require a password).

```
[paths]
default = ssh://[uid]@cycle1.csug.rochester.edu//u/cding/cs253hg
```

Set up your user name by creating the file in your **home directory** */.hgrc* and add the following. Note this is not the same file as *.hg/hgrc* inside *cs253hg*.

```
[ui]
username = First, Last <email>
```

Now you are ready to connect to *cycle1.csug* and download the content. Inside *cs253hg*, use the following two commands first to download the content into the local repository and then check out the current version of files.

```
hg pull
hg update
```

After the two steps, you will see a copy of this draft book in *cs253hg/dynbook/*.

1.8 Core Material

The core material falls in four categories:

- *Languages*: lambda, iterator, mix-in, meta-class, type, stream, lazy evaluation.
- *Tools and processes*: distributed version control, test-driven dev, code coverage, test generation, type checking, teamwork, code review, software process and lifetime.
- *Design and construction*: design principles, design patterns, component and composition.
- *Dynamic systems*: Ruby interpreter, Jruby, web server, memory and caching, other systems such as WebWork.

1.9 Course Outline

The content will be added after the lecture.

Chapter 2

Dynamic Languages

2.1 Pre-lecture Preparation

1. Take a look at the on-line TIOBE programming community index at http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html.

   - How does it quantify usage or popularity?
   - Of the top 20 languages, how many of them are dynamic languages? What makes them dynamic?

2. Try downloading Ruby to your desktop or notebook computer. You need the interpreter ruby, the interactive shell irb, and the package manager gem. You can also download JRuby, which is a more recent implementation based on Java. (Ruby will be the language used in class instruction and demonstration, but students are free to use other programming languages in assignments).

   - Try the following in irb. It should take no more than a few short lines of code.
     - Write a hello-world function
     - Define a class and create an object
     - Inspect files in a directory
     - Connect to www.cs.rochester.edu
     - Open a dialog box on the screen (JRuby)
     - Open a page on your browser
   - Can you compose a program interactively?
   - Can you write self-inspecting, self-modifying code?

2.2 Main Features

The content will be added after the lecture.

2.3 Dynamic Code Inspection and Construction

The content will be added after the lecture.

2.4 More Complex Operations

The content will be added after the lecture.

2.5 Language Popularity Index

The content will be added after the lecture.

2.6 Exercises

1. Can you extend the Ruby Array class to add similar support as in R to construct a continuous number sequence and to index through an array?
2. Compare the supported operations for arrays in other languages.
3. An array may be large, so it is useful to have a search method. In Ruby, we inquire whether an array supports a method such as *grep*. Let's take a leap of faith and try *grep* on the method array. Running `Array.public_methods.grep ‘‘grep’’` you will find the answer is no. The returned array is empty. But this seems contradictory. Haven't we just used *grep* on an array?

Bibliography