The Java Programming Structure
IFT 194: HW 1

Brandon Doyle
bdoyle5@asu.edu
1215232174

Dr. Usha Jagannathan
Usha.Jagannathan@asu.edu

July 4, 2018

Reading Questions

1. According to the reading, the original name of the Java programming language was "Oak."
2. The author of the reading, Michael O'Connel made the claim that "... the solution preceeded the problem" regarding the creation of Java. In this statement, the "problem" arose in mid-1994 when the Internet's popularity began growing. As Gosling says, there was a need for a "really cool browser," which is an application that should be architecture-neutral, real-time, and secure. These are all qualities that had been built into Java since 1991.
3. The creator of Java is James Gosling. The author also mentions Patrick Naughton, who was the "project lead on Sun's OpenWindows user environment before joining the secret Green team," which was the team name for a consumer electronics effort at Sun Microsystems in the 1990s. Gosling's contributions to the team were usually to solve "tooling" problems that required extensive programming. Gosling had already, by that time, produced a number if interesting projects, such as the first implementation of the Emacs text editor in pure C (I actually have GNU's Emacs installed on my system now).
4. The creators' motivations in writing Java were to make the language platform-neutral for consumers. Today there are a large number of different processors and architectures, even more-so than there were in the '90s. For example, the text also states that the goal was to "build a system that would let us do a large, distributed, heterogeneous network of consumer electronic devices all talking to each other."
5. Java was spun into the web when Jonathan Payne wrote a web browser called WebRunner in Java. This allowed developers to write applications in the browser in Java.
6. The original target market for Java was consumer electronics. Java was created to solve the development issues programmers were having with creating systems running on different platforms (operating systems and architectures) that could communicate with each other. Languages (like Scala) that run on the JVM are platform independent.

Textbook Exercises

1.14

According to the text and the Java SE10 language specification, there are two types of comments, including

1. end-of-line comments, denoted by '`//`', and
2. traditional comments, denoted by '`/*...*/`'.

End-of-line comments delimit text on a single line that is to be ignored by the compiler, and traditional comments, or multiline comments, can mark several lines of text that are to be ignored.

View the source of this document on GitHub.

1.15

We are tasked with determining which of the following identifiers are valid and invalid.

1. `Factorial` – This is a valid identifier. However, it should be used as an identifier for classes or types, since it begins with a capital letter.
2. `anExtremelyLongIdentifierIfYouAskMe` – This is a valid identifier. Again, referencing the Java SE10 specification, identifiers have an unlimited length limit. This has been a feature since at least Java 6.
3. `2ndLevel` – This is an invalid identifier because it starts with a number.
4. `level2` – This is a valid identifier.
5. `MAX_SIZE` – This is also a valid identifier.
6. `highest$` – This is again a valid identifier, because '`$`' is a valid special character in identifiers.
7. `hook&ladder` – This is invalid, since '`$`' is not a valid character in an identifier, according to the language specification.

Actually, interestingly enough, the specification states the following, which I thought was pretty interesting.

> The "Java letters" include uppercase and lowercase ASCII Latin letters A-Z (\u0041-\u005a), and a-z (\u0061-\u007a), and, for historical reasons, the ASCII dollar sign ($, or \u0024) and underscore (_, or \u005f). The dollar sign should be used only in mechanically generated source code or, rarely, to access pre-existing names on legacy systems. The underscore may be used in identifiers formed of two or more characters, but it cannot be used as a one-character identifier due to being a keyword.

1.16

We are tasked with determining, based on the text, which of the following are good identifiers.

1. `q` – This is a poor identifier.
2. `totVal` – This is a mediocre identifier,
3. `theNextValueInTheList` – This