APCS Problem Set 1: Basics of Computing

Michael Ferraro
mferraro@balstaff.org
Balboa High School
25 August 2015

Contents

1 Introduction

Welcome to your first APCS problem set! You should expect 3-4 problem sets every six-week marking period. It is expected to take you quite a few hours to complete each one, so plan accordingly -- **don't procrastinate!**

1.1 Why Problem Sets?

First, since an AP course is supposed to approximate a college experience, I think it's important to assign you work in much the same way a college-level computer science class would -- by way of problem sets. Aside from

checkpoints in each problem set, to make sure you're on track to complete it on time, you get to choose your own pacing, but you are ultimately responsible for having completed the problem set by the due date.

Second, by giving you a problem set, I'm giving you a view, from start to finish, of what I expect for you to learn over the coming days. **When you get a new problem set, scan through it before you begin.** Make it a *goal* to learn all you need to in order to accomplish the tasks I set up for you. And **when you don't understand** what's being taught or what you're reading, **make sure you come in for extra help.**

1.2 Objectives

By completing this problem set, students will learn...

- the function of components in a computer;
- the fundamental roles of operating systems;
- how data is represented and stored; and
- the process of writing, compiling, and running simple Java programs.

1.3 Due Date

This problem set is tentatively due **before 5th Period** on *2 September 2015*. **Due dates in problem sets will always be approximate** -- and more tightly defined closer to those dates -- for these reasons:

- I need to have the flexibility to replan lesson sequence, duration, and content on the fly in order to adjust to the class' reaction to material.
- It's hard to plan specific dates when there are fire drills, unanticipated school closures, unplanned special schedules, etc.

Rarely (if ever) will a due date be earlier than what is stated in the problem set. Final due dates will be announced in one or more of these places: Course email list, lesson slides, and verbally in class.

2 Intro to Computer Hardware and Software

2.1 Required Reading

As stated in the syllabus, don't skip the required reading! The reading generally isn't many pages and is an excellent supplement to the lessons given in class. Remember: Taking the time to read the textbook will *save you time* as you answer questions from the textbook![1]

→ §☐ of this problem set (but don't sign until later)

→ `Litvin & Litvin, Java Methods A & AB: §§1.1-1.4`: These sections are available on the publisher's website at http://www.skylit.com/javamethods/.

2.2 Define Terms

In the textbox below, copy and paste this text -

```
* CPU:

* RAM:

* GUI:

* Terminal Shell:

* Algorithm:
```

- and supply definitions to the right of each term.

2.3 Computer Boot Process

A computer's ROM (usually referred to as the *BIOS*[2]) holds instructions on how to communicate with the hard disk and find a boot record on it. That boot record holds instructions on how to find and start the

.

2.4 Peripherals

List three examples of peripheral devices, separated by commas.

save progress

3 Representing Information Using Computers

3.1 Required Reading

→ Litvin & Litvin, *Java Methods A & AB*: §1.5[3]: This section is available on the publisher's website at http://www.skylit.com/javamethods/.

3.2 Binary as Necessity?

Why **must** computers represent information in binary form?

3.3 Conversions

3.3.1 Binary to Decimal

Conversions from binary numbers to their decimal equivalents are included in the book questions in §. You might skip to Ch. 1, #12 in that section before proceeding to the next subsection, where the more difficult decimal-to-binary conversions are required.

3.3.2 Decimal to Binary

For each of the following, convert the given decimal number to its binary equivalent.

1. 6 =
2. 16 =
3. 33 =

3.4 Book Questions

1. Ch. 1, #10.
   10a)
   10b)
   10c)
   10d)
2. Ch. 1, #11.
   11a)
   11b)
   11c)
3. Ch. 1, #12: Supply the value for the indicated column.
   12a) ← Decimal value
   12b) ← Binary value
   12c) ← Decimal value
   12d) ← Binary value
   12e) (skip)
   12f) ← Decimal value
   12g) ← Decimal value
4. Ch. 1, #12, **hex** column only.
   12a)
   12f)
5. Ch. 1, #13: Two hints are given below. Use whichever you find easier.
   Hint A: If one particular outcome is HHTHTTHTTH, you might think of that being like the binary sequence 0010110110. This question is really asking how many binary sequences are possible given 10 bits.
   Hint B: Consider games with fewer tosses and draw a tree of possible outcomes. Find a shortcut for computing how many outcomes there are if you know the number of flips.
6. Ch. 1, #16: Be sure to show sample tic-tac-toe games and the way in which you'd encode that game's state (or the arrangement of Xs and Os). *Hint: Imagine that you were going to take a mental snapshot of a tic-tac-toe game you see. What information would you have to record in order to recreate that exact setup again later? How could you use a string/sequence of 0s and 1s to ``save'' that information?*

   Complete this problem on the paper form.

save progress

4 Basics of Software Development

4.1 Required Reading

→ Litvin & Litvin, *Java Methods A & AB*: §§2.1-2.3

4.2 Software Life Cycle

§2.1 has a bulleted list of tasks involved in software development, which represent the *software life cycle*. Take that list and present it here as a flowchart, where each box is one of the steps Litvin mentions. Using arrows, connect each box to the box or boxes that represent the next logical step(s) in the process.

To help you get started, refer to the image at http://feromax.com/apcs/problemsets/PS01/downloads/flowchart.png.

Complete this problem on the paper form.

4.3 Bugs!

Explain the origins of the term *bug*.

4.4 Code Reuse

The idea of *code reuse* is very important in software development. For example, an engineer at Oracle may have written a function called `sqrt()` in the `Math` package that's included in Java for you and other programmers to use. A good reason for you to print the square root of 17 in this way --

```
System.out.println( Math.sqrt(17) );
```

-- as opposed to writing many more lines of code that actually compute the square root of 17 through a complicated set of steps is that you save lots of time. And you can rely on `Math.sqrt()` giving you a correct answer since it's been thoroughly tested by others -- as opposed to code you might dream up that might not work right.

Based on the required reading and what's written above, answer the following.

1. Professional programmers prefer to write their own code even if the programming languages they use already have built-in functions that do the same thing.
2. For code to be considered reusable, it has to be tested under all possible conditions so those using the code can be reasonable certain that it'll work for them.

4.5 Book Questions

1. Ch. 2, #1.
   1a)
   1b)
   1c)
   1d)
2. Ch. 2, #2: List the four languages, separated by commas.

3. Ch. 2, #3: If false, indicate why. (No need to explain true statements.)
   3a) [ ] if false, explain why below.
   [ ]
   3b) [ ] if false, explain why below.
   [ ]
4. Ch. 2, #4: [ ]

save progress

5 First Programs

By now, you've created various versions of `HelloWorld.java`. You've also learned how to do some basic integer math with Java. Now you get to be creative!

5.1 How old were you then?

Write a program that asks the user to provide his/her name and age (separately, using a `Scanner`)[4] and then outputs the following: ``Wow, <*name*>, I had no idea that you were <*formerAge*> years old three years ago!" Note that *formerAge* is 3 years less than the current age.

Once you're done, you'll need to demonstrate to your teacher that your program works *as specified* (you'll hear about *specifications* in software later).

Get teacher's initials on paper form.
Complete this problem on the paper form.

5.2 Simple Math

In the space below, write code that answers Ch. 2, #11. Note that you must write the program on a computer, first, before submitting your written response. That will help you to correct errors that might otherwise prevent your program from working.

***** IMPORTANT *****
You will need to save your source code to your GitHub repository! You will be instructed in class how to initially push your `ps01` project to GitHub. Once pushed, you'll be able to update your uploaded solution from school or home until the problem set due date.

When writing code by hand -- as you'll need to do on the AP exam -- make sure you write neatly and show proper indentation. Points will be deducted for hard-to-read responses.

Complete this problem on the paper form.

save progress

6 Conclusion

6.1 Required Reading

→ Litvin & Litvin, *Java Methods A & AB*: §2.7[5]

6.2 Checking for Understanding

1. When writing a program in an interpreted language, like Python, what step does the programmer get to skip - unlike when programming in Java?
2. An IDE is an example of a GUI application.
3. Object-Oriented Programming tends to lower development costs.

save progress

7 Submission & Affirmation of Academic Honesty

Complete this problem set on (or before) the due date mentioned in 1.3. All work - typed and handwritten - must should be *neat*. Remember, spelling and grammar are considered during the grading process!

You should recall the section in the course syllabus titled *Academic Honesty*. **If you have complied with the guidelines set forth there, please complete the portion of the paper form corresponding to this section of the problem set.**

> ``*I hereby certify that I have upheld the guidelines for Academic Honesty for this course. All work presented here, in written form, and in my APCS locker, in file form, is my own. Students with whom I worked and/or consulted for help are listed below."*

Footnotes

... textbook![1]
: Your teacher often refused to read his textbooks back in high school and college. Had he acquired the habit of reading -- especially his computer science textbooks -- he would've had the chance to practice reading code showing examples of the topics that were discussed in class, and thus would have performed much better in his classes!

...*BIOS*[2]
: See a good definition of *BIOS* at http://www.thefreedictionary.com/bios.

... §1.5[3]
: When reading §1.5.1, it's not necessary to understand all of the details about how signed and unsigned numbers work, nor is it necessary to understand the floating-point number representation. The AP exam is unlikely to ask for specific details about these topics.

...Scanner)[4]
: You may want to look at the *More Greetings* example in §2.4 to see how to prompt a user to provide input on the command line.

... §2.7[5]

The last paragraph is about *OOP*, which will be covered in the next lessons.

Michael Ferraro 2015-08-23