Computer Science 210

Data Structures
Summary

• Today
  • In-class work on Java: Gnome
  • Static data and methods
  • Compiling and running Java
  • main
  • Arrays
  • Input and output
  • for loops
  • break, continue
  • Writing a (Java) program

• Examples
  • Gnome, CreditCard

• READING: GT chapter 1, 2
Writing a (Java) program

1. Design.
2. Come up with pseudocode or flowchart.
3. Write code.
4. Test and debug.

- People mistake programming with step 3.
- People mistake computer science with step 3.
- I’ll make sure you won’t.
Writing a (Java) program

1. Design
   - the most important step
   - you design your world, model your classes, assign responsibilities and behavior
   - how things will work and who will do what
   - Guidelines
     - responsibilities/encapsulation:
       - each class has a different job
     - independence:
       - each class should be as independent from others as possible.
       - Each class should be autonomous over some part of the world.
   - You can create the world any way you want it. You are the God of your world.
   - But...Keep in mind YOU will implement this world and try to make it work.
   - Design your world so that the structure and the interactions are clear & natural.
Writing a (Java) program

2. Pseudo-code

• pseudo-code is a mixture of code and English
• no real guidelines, just that it should be clear and precise enough so that somebody who knows the programming language can get it to work without much effort
• you use pseudo-code to write down the algorithm/logic of your code, without the tedious Java details

• While writing pseudo-code you may go back to your world and change it, to make it simpler.

• When you’re done with design and pseudo-code
  • you’re done with the hardest part
  • hopefully your world is flawless
  • now you just need to make it work
Writing a (Java) program

3. Coding

4. Testing and debugging
   • add features incrementally
   • test and debug
   • DO NOT write more code than you can debug.
   • YOU will have to debug your code.
   • Debugging:
     • use print statements
     • use debugger
Readability and Style

- Use meaningful names; use constants
  - [see textbook on style guidelines]
  - [see link]
- Write small methods
  - if a method is longer than one screen, break it into sub-methods
- Commenting
  - How much commenting? Your code and its comments should be such that anybody can take a look and understand how your world works.
  - Nice to get used to Javadoc style.
  - WRITE COMMENTS AS YOU CODE, do not leave it for “later”.
  - Commenting has to become your second nature. COMMENT FOR YOU.
  - if your code is not commented, no good style, YOU will have a hard time getting it to work.
- Programming should be fun!
  - make it fun by following these guidelines.
Readability and Style

- Encapsulation
  - objects should interact with each other knowing only their interface; a class does not need to know the IMPLEMENTATION details of other classes

- Independence
  - make each method/class as independent as possible. Make as few assumptions as possible.

- Structure
  - Never take shortcuts at the expense of clarity.
  - Never optimize at the expense of clarity.

Simplicity
Clarity
Generality
For next time

• Reading
  • read textbook chapter 1 and 2
  • see class website links on writing clear code

• Exercises
  • Gnome
  • loops

• Code examples
  • Gnome
  • CreditCard
  • Scanner