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
    • in-class work: for loops
  • break, continue
  • writing a Java program

• Examples
  • Gnome, CreditCard

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

• 1. Design
• 2. Pseudo-code
• 3. Coding
• 4. Testing and debugging

• 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.
• Goal: design your world so that the structure and the interactions are clear and 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
  • comment
  • small methods
    • if a method is longer than one screen, break it into sub-methods
Writing a Java program

• More 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

  • Your code and its comments should be such as anybody can take a look and understand how your world works

  • Simplicity
  
  • Clarity
  
  • Generality
For next time

• **Reading**
  - see class website links on writing clear code
  - read textbook chapter 1
  - GT chapter 2

• **Exercises**
  - Gnome
  - loops

• **Code examples**
  - Gnome
  - CreditCard
  - Scanner