CS 5044
Object-Oriented Programming with Java

Q&A Session

Welcome!

- Keys to success:
  - Frequent interaction via Piazza
  - Manage your time each week
  - Minimize procrastination on assignments
- Problems?
  - Generally, due dates cannot be extended
    - Please contact me as soon as practical if you think an exception might apply
  - You have 96 hours in a Time Bank for **Project** assignments only (see the Syllabus)
    - You may use up to 48 hours per Project assignment
    - Manage these hours as you wish; allocate them at your discretion
  - Please let me know about any special accommodations you may need

Academic integrity (Graduate Honor Code)

- **All assignments are to be done entirely individually** (no group work!)
  - No help to/from any others permitted, including tutors, past/current students, web sites, ...
    - **One Exception**: Collaboration via our Piazza forum is very highly encouraged!
  - Do NOT make your code available to others, including via public repositories
- All Piazza forum postings are safe (unless you've received a specific warning)
  - For purely practical reasons, please don't post code, unless asked by an instructor
  - Feel free to answer questions asked by your classmates
  - If you post too much, we reserve the right to edit or remove your posting
    - We may also mark private postings as public, with or without editing
- Plagiarism includes copying of source code, not just narrative documents
  - Cheating occurs in many different forms; all of these will be pursued
- Please contact me as soon as possible, if you:
  - ...have any questions about what is allowed by the GHC or course policy
  - ...feel you need any external resources, or to seek any help from others
  - ...received materials from former/current students; you must return or destroy these
  - ...are lost/struggling to get started; we may need to re-assess your readiness
  - ...become overwhelmed by the workload; we may need to explore your options

About the assignments (all due Mondays at noon ET)

- Graded discussions:
  - It's generally easy to add to the discussion; try to interact with your classmates
- Readings:
  - You aren't expected to read every word of every section, example, sidebar, etc.
  - Decide for yourself what depth is appropriate, given your background and learning style
- Quizzes:
  - The course tends to place an emphasis on the specifics at the foundation of Java
    - There are no "trick" questions (intended to deceive) but read everything carefully
  - Programming is fundamentally a detail-oriented task
    - Some students tend to experience great difficulty and/or frustration with this
- Projects and Homeworks:
  - Time needed to develop software varies **extremely widely** from student to student
    - For example, a programming task that takes an *average* (across all students) of 10-15 minutes to develop might sometimes actually take well over an hour to complete!
      - This is just due to natural variation; it's not an indicator of poor learning or low quality
      - However, we must strive to produce code in a reasonable amount of time
  - Try to allocate extra time for unexpected issues that can arise during development
    - A single typo may introduce a bug that can take a very long time to find and fix

Module 1 reading highlights

- Java is a cross-platform, statically-typed, object-oriented language
  - Every variable must have a declared type, which is enforced by the compiler
    - Every value also has an inherent type enforced by the compiler
  - **Classes** are the fundamental units of development; **objects** are instances of classes
    - All the code you write is located within some class, which itself represents a type
    - However, there are also primitive types, which are not classes
- Classes typically:
  - Contain *private* **instance variables** (often called **fields**) that hold state information
  - Expose *public* **methods**:
    - Methods accept zero or more parameters as input, each of an explicitly declared type
    - Methods optionally provide one return value as output, of an explicitly declared type
      - The special type `void` indicates that there is no return value
    - We generally categorize methods into one of two kinds, based on their behavior
      - A **mutator** *potentially* alters the state of the object on which it's called
        - » May or may not provide a return value
      - An **accessor** will never alter the state of the object on which it's called
        - » Typically provides a return value to be meaningful
      - None of this is enforced by the compiler; it's important to follow naming conventions

Primitives and objects

- Some (very important!) notes regarding variable assignments (Horstmann 2.8)
  - Each **primitive** type variable holds a single *value*
  - Each **reference** type variable holds a pointer ("reference") to a single *object*
    - Multiple reference type variables can point to ("reference") the same object
    - We'll also explore how reference variables can be *null* (pointing to no object at all)
- Example: Primitive variables

```
int x = 3;
int y = x;
```

  - Each of these variables contains a single, independent, primitive value
  - Primitive variable values can only change via re-assignment to a new value

```
x = 5;                          // This changes x, but does not affect y
```

- Example: Reference variables

```
BankAccount a = new BankAccount(100);
BankAccount b = a;
```

  - Both of these variables now point to the same object instance
  - Object state can be mutated via method calls; the `new` operator creates a new object

```
a.deposit(50);                  // Any state change affects both b and a (there's only one object here!)
a = new BankAccount(5000);      // This assignment does not affect b (now there are two separate objects)
```

Project 0 hints and tips

- Use `println()` rather than `print()` to generate the output
  - Calling `println()` appends a line terminator appropriate for the runtime platform
  - Don't hard-code line terminators that happen to work on your development platform!
- Copy-and-paste the desired output from the assignment page to your code
  - This can avoid lots of potential problems with misalignment and other spacing issues
  - Use 6 separate lines of code; you don't need to concatenate the output into one line
- PLEASE: Import the style Preferences into Eclipse (see Installing Eclipse)
  - This will save you lots of time throughout the semester
  - Use Ctrl-Shift-F (or Cmd-Shift-F) to reformat your source code accordingly
    - Points are automatically deducted if you don't follow these style rules!

Project 0 demonstration

- Eclipse walk-through...