Overview

- Essential Java Topics
- How Web Apps Work
- Internet of Things
- CS Past 15

Essential Java Topics

visibility modifiers | final & static keywords | parameter passing

Visibility Modifiers (1/2)

- Can apply to classes, methods, and instance variables
- **Public** can be accessed from any class – use for methods (not variables) that you want to expose, i.e., all but a few **private** helper methods
- **Protected** can be accessed from a subclass or any other class in the package (can’t apply to classes, just methods and variables)
  - Java unfortunately doesn’t have an access level which only allows the subclass access
  - Some of you discovered protected while doing Tetris. We do not encourage its use because it violates encapsulation principles (we’ll take off if we see it in final projects)

Visibility Modifiers (2/2)

- Default or package protected (no access modifier) can be accessed from any class within the package
  - **Private** can only be accessed within the class and any inner class

Instance variables can be accessed using the same “dot notation” as methods:
- **Public** instance variables can be accessed using the same “dot notation” as methods:
  - Violates encapsulation as well, and not allowed in CS15
- **Protected** can be accessed from any class within the package

What’s so bad about **public**?

- **Public** instance variables can be accessed using the same “dot notation” as methods:
  - Violates encapsulation as well, and not allowed in CS15
- **Public** instance variables are bad for a reason!
Final Classes and Methods (1/2)

- `final` keyword can be used for classes, methods, and variables
- Similar meaning for classes and methods, but different for variables
- A `final` class cannot be subclassed and a `final` method cannot be overridden

```
public final class HelloWorldClass {
    // some code
}
```

```
public final class Example {
    public final void reallyComplicated() {
        // some code
    }
}
```

Final Classes and Methods (2/2)

- Whenever a class is not intended by the programmer to be subclassable then it should be `final`
- Ex: `final` Ex: `final` Ex: `final`
- A method should be `final` if it is integral to the proper functioning of the class
- Prevents someone else from incorrectly trying to extend a class or override a method

```
public final class HelloWorldClass {
    // some code
}
```

```
public final class Example {
    public final void reallyComplicated() {
        // some code
    }
}
```

Rule of thumb: Design for inheritance or prohibit it!
* Source: Effective Java (Joshua Bloch)

Final Variables (1/2)

- You have used `static` `final` variables when declaring constants
- A `final` variable can only be assigned a value once and cannot be reassigned
- Must be assigned by the time the object has been constructed

```
public class Example {
    public final int myInt;
    public final Point myPoint;
}
```

```
public class Example() {
    myPoint = new Point(1,2);
    myPoint = new Point(3,4); //okay
    myPoint = new Point(5,7); //error
    //other error: myInt not assigned
}
```

Final Variables (2/2)

- If the variable is a reference to an object, the variable cannot be changed but the object can still be mutated
- If you know that a variable should never be changed, good practice to make it `final`
  - Ex: `final` Example

```
public class Example {
    public final int myInt;
    public final Point myPoint;
}
```

```
public class Example() {
    myPoint = new Point(1,2);
    myPoint = new Point(3,4); //okay
    myPoint = new Point(5,7); //error
    //other error: myInt not assigned
}
```

Rule of Thumb: Minimize Mutability!
* Source: Effective Java (Joshua Bloch)

Static Methods (1/2)

- You've already seen static methods. Ex:
  - `Math.random();`
  - `Color.red();`
- `Static` methods are invoked on a class, not an instance
  - No need to ever call "new `MyClass()" etc.
  - Also means that they cannot access instance variables, since those could be different for each instance

```
public class Example {
    public final int myInt;
    public final Point myPoint;
}
```

```
public class Example {
    public final int myInt;
    public final Point myPoint;
}
```

Static Methods (2/2)

- You probably haven't written a static method yourself so far
  - `static` visibility: `static` method type (parameters) { }
- When should you make your method `static`?
  - Following two conditions should be met:
    - It doesn't modify program state
    - the result that you are returning depends on nothing but the parameters that you are taking in
  - Ex: Say you were writing a `Geometry` class that helps you do math with points

```
public static double getDistanceBetweenPoints(Point2D p1, Point2D p2) {
    // some code
}
```

```
public class Example {
    public final int myInt;
    public final Point myPoint;
}
```

```
public class Example {
    public final int myInt;
    public final Point myPoint;
}
```

```
public class Example {
    public final int myInt;
    public final Point myPoint;
}
```

```
public class Example {
    public final int myInt;
    public final Point myPoint;
}
```
Progression in scope:

- Local variables are known in a single method
- Instance variables are known to all methods of a class
- Class variables are known to all instances of a class (sometimes referred to as class variables)

If we want all instances of a class to share the same value for a variable, declare it `static`

```java
public class CSS15Student {
    private static int _numStudnets = 0;
    public CSS15Student() {
        ++_numStudents;
    }
    public static int getNumStudents()
        return _numStudents;
    }
}
```

From outside the class, we can call `System.out.println(CSS15Student.getNumStudents())` to print the current number of CSS15Student instances.

Be careful with these!
- Not very object oriented - violates object encapsulation
- Cannot be garbage collected

Therefore, generally only use in combination with `final` for constants

```java
public class CSS15Student {
    private static int _numStudnets = 0;
    public CSS15Student() {
        ++_numStudents;
    }
    public static int getNumStudents()
        return _numStudents;
    }
}
```

From outside the class, we can call `System.out.println(CSS15Student.getNumStudents())` to print the current number of CSS15Student instances.

Methods can take in any data type as a parameter
- Primitives such as `int, boolean, char`
- Objects
- How are data types passed into methods?
  - Mechanics of parameter passing
  - How do modifications made to the arguments inside the called method affect the data in the caller method?

### Declaring Variables

- What actually happens when we declare a variable?
- Variables are pieces of memory that hold a value
  - Primitive variables hold primitive values
  - Object variables hold the memory address of the Object
- When a variable is declared, memory is allocated for the variable’s value
  - Creates a small “box” of memory for a value to be put in later

```java
/* Primitive */
int x;
/* Object */
Dog dog;
```

### Initializing Variables

- What actually happens when we initialize a variable?
- For primitives, the value is placed in the variable’s memory
- For Objects, the Object’s memory address is placed in the variable’s memory
  - When an Object is instantiated, memory is allocated for the Object, can be quite large
  - A variable assigned to that Object will store the Object’s memory address

```java
int x;    // Primitive
x = 10;   // Primitive
Dog dog; // Object
dog = new Dog();     // Object
```
Primitives in Java are “passed by value”:
  - changes to value of parameter only persist through scope of method
  - When a primitive variable is passed to a method as an argument, a copy is actually passed
    - modifying copied parameter within callee will not affect on contents of caller’s variable
  - What will these print lines print out?

Objects in Java are also “passed by value”:
  - value is memory location of object, not object itself -> no copying entire object!
  - When object’s variable is passed to method as argument, variable’s value is still copied, but what is copied is a memory address
    - modifying object within callee will have effect on caller’s object
  - What will these print lines print out?
  - Note: this is only Java’s definition of pass by value!

There are languages that “pass by reference”:
  - variable passed is not copied, meaning that reassigning the variable in the callee will change the object that the caller’s variable points to as well
  - However, in Java, reassigning the variable in the callee will not change the variable in the caller
  - What will these print lines print out?
  - in a pass by reference language, the second print line would read “Winnie”
  - dog would be pointing to the new Dog instantiated in Dog(name)

There are languages that “pass by value”:
  - in a pass by value language, the second print line would read “Winnie”
  - dog would be pointing to the original Dog object
Returning Objects

- Returning a variable pointing to an object (as in an accessor method) is similar to passing one as a parameter
  - the caller of an accessor method will be able to directly modify the object it returns
  - other objects will be able to modify private instance variables, without a mutator method
- Preserve encapsulation and prevent unwanted access by returning copies of objects in accessor methods

Let’s look at an example webpage

A.K.A. The World Wide Web
(but really just how web apps works)

Two Main Parts

- What the user sees & Where the data comes from

  Two common ways of describing this:
  - “frontend” vs. “backend”
  - “client” vs. “server”

Frontend – What the user sees

Three major elements go into displaying many webpages:
1. HTML -- Content
2. CSS -- Style
3. JavaScript -- Interaction

Frontend – Part 1 (HTML)

- HTML: Hypertext Markup Language
  - The most foundational part of a webpage
  - Broken down into subdivisions
    - Like a lecture deck that has slides, with bullets, each with subbullets
    - HTML relies on tags to describe different parts of the page represent
      - e.g. “This is a header,” <h1>” or “This is a paragraph,” <p>”
    - Tags act as cues for styling sheets to understand how to style the page
    - Contrast with old procedural methods using markup to specify formatting (e.g. \[ to insert italic text)
In 1967, Andy, Ted Nelson (who coined the term “hypertext”), and then-student Steve Carmody co-designed Hypertext Editing System. At the same time, Doug Engelbart & 30 full-time researchers developed the NLS (“oN-Line System”) at Stanford Research Institute. Debuted it in the “Mother of All Demos” in late 1968. Displayed outline text, the mouse, hyperlinks, and many other features that led to Xerox PARC’s developments and, from there, to Apple and Microsoft.

Brown’s Hypertext Editing System was the first hypertext system on commercial equipment (Brown’s IBM/360 model 50, under IBM sponsorship). Was used by professors and by NASA to produce documentation for the Apollo program (on microfilm the astronauts had with them).

CSS: Cascading Style Sheets

- The dressing on the webpage, responsible for styling
- Can apply “styles” to specific elements or types of elements on the page
  - “Turn the text of all paragraphs blue”
  - “Add a red border around the element that I called #myBoxWithARedBorder”
- Connect HTML pages to style sheets to format them
- Webpages with just HTML and CSS only are meant to be “static,” so generally don’t have much complicated user interaction.

Let’s add some styling to our webpage from before:

```html
Hello World!
</body>
</html>
```

Attach the CSS file to the HTML

Attach the CSS file to the HTML

```html
body {background: gray;}

#myBox {color: blue;}

#myBoxWithARedBorder {border: red 2px;}

#myParagraph {font-weight: bold;
```

```html
</head>
</body>
</html>
```

What controls the bulk of the user interaction and data manipulation on the user end?
- This is one way of "scripting" and writing more complicated code on the frontend, but JavaScript is the most common.
- Many frameworks (such as Angular.js or React.js) are built on top of JavaScript.

Code that is run in the user’s browser
- Example uses:
  - Decrease a countdown timer on the page every second
  - Sorting functionality on websites (e.g. order by price)
  - The Google Sheets TA Hours signup sheets

JavaScript is often what asks the server for data, and then organizes what it gets back for the user.

Entirely unrelated to the Java that we know
- JavaScript is partially OOP, but is messy and easy to get wrong
- More flexible than Java, but much more finicky.

Because all frontend code (HTML, CSS, JavaScript) happens in your browser, you can see the code for any website that you want!
Making pretty websites is great, but how does the website do data-related things (e.g. passwords, searching a movie database, etc.)?

- The backend is where the bulk of data processing happens
- Backend is what interacts directly with the database
- Java is often used as a primary backend language, as is Python (which is taught in CS16)
  - But there are many dozens more that are commonly used

**Backend – Where the data comes from**

Databases store information in an organized manner
- Makes it easy and quick to query for information
- Can be queried in many different ways to get different subsets of information
- “Relational databases” are most common, such as SQL (“Structured Query Language”)
  - Consists of many tables that relate to each other
  - For example, might have an accounts table and a passwords table, so a user in the accounts table also has a row in the passwords table
  - Example SQL query:
    ```
    select first_name from users where last_name = "van Dam"
    returns
    ["Andy", "Bjorge"]
    ```

**Backend – Databases**

Websites have names that are easy to remember, like www.brown.edu
- When you type a website name to your browser, your Internet Service Provider (ISP, like Verizon or Cox) looks at a Domain Name Server (DNS)
  - DNS returns an IP address of the server to connect to
  - DNS is often called the “phonebook of the internet,” since you can look up IP addresses from a website name
- Can connect to http://www.brown.edu or http://128.148.252.151/ in your browser and get to the same page

**Getting to the Webpage – IP Addresses**

Combination of backend & frontend technologies is often referred to as “technology stack”

**Putting It All Together (1/3)**

Let’s walk through one way the cab.brown.edu site might work:

1. You type cab.brown.edu into your browser
2. Your Internet Service Provider gets the IP address of the brown.edu web servers and connects you
3. Your browser gets HTML, CSS, and JavaScript from the server to load in your browser
4. You search for “Computer Science” courses by clicking on the drop-down, created by the HTML and styled by CSS

**Putting It All Together (2/3)**

5. The JavaScript registers the event, and sends a request to the servers (previously located in the basement of the CIT), asking for information
6. The backend code registers the request, and queries the database for all Computer Science courses
7. The database returns a list of all CS courses for the backend code to process
8. The backend code returns the list, after any data processing, to the frontend (your browser)
9. The JavaScript in your browser updates what courses are displayed on the screen

**Putting It All Together (3/3)**
What do the buzzwords mean

<table>
<thead>
<tr>
<th>Communication</th>
<th>Things</th>
<th>Internet of Things</th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection and exchange of data</td>
<td>Any physical object, from devices (phones, laptops, etc) to cars to buildings to tables, and so on</td>
<td>Embedding things with electronics, software, sensors, actuators, and network connections to allow communication</td>
</tr>
</tbody>
</table>

How does it work?

- Embed “things” with microcomputers
  - Raspberry Pi
  - Arduino
- Connect microcomputers to sensors
  - Cameras
  - Temperature/Humidity/CO2 sensors
  - Accelerometer/Gyroscope sensors
- Connect microcomputers to Internet
  - Wifi chips (can be unreliable)
  - Ethernet
- With internet, can send data and receive data

What can you do with some sensors and the Internet?

- Smart Homes
  - Smart Mirror
  - Facial Recognition Door
  - Nest Energy Saver
  - Air Quality Egg

- Environmental Studies
**Health Applications**

- mimo monitors: help prevent SIDS

**Mind Control**

- Jack Stewart flew a plane with his mind by concentrating on specific arrows on a screen.
- Electrical signal is created when green box focuses on desired command.
- Sensors pick up that signal, send message to computer, computer responds accordingly.

**Dangers of a “More Connected World”**

- **Privacy**
  - IoT depends on collecting and sharing data
  - What data can be shared? What data can’t?
- **Digital Surveillance**
  - Smart Billboards
  - Target knew my daughter was pregnant before I did
  - By analyzing network traffic patterns, can tell when people are home/not home
- **Security**
  - “Hacking” becomes physical as well as virtual
  - Can tamper with physical devices and gain access to all connected devices on the network

**A “More Connected World”**

- Up to us as future programmers to decide what that means
- **Interfaces**
  - “If users need to learn different interfaces for their vacuums, their locks, their sprinklers, and their coffee makers, it’s tough to say that their lives have been made easier.” – Michael Littman, “The Path To A Programmable World”
  - How will these applications change our daily lives?
  - Not just phones/tablets - bands, medical devices, home devices, car devices, etc.
- **Data-driven decisions**
  - How will these changes affect our day-to-day decisions?
  - What data is “safe” to use, what needs user consent, what data shouldn’t be shared?
- **Poses general question:** how can technology make our daily lives easier/better?
  - Very new field — Get to define what IoT is, how it’s used
  - Explore how different fields in CS can be applied to IoT
  - How can we interface with technology to make our lives easier/better?

**Other Mini-Lectures**

- In past years, HTA lectures have covered other topics in CS
- Slides from those lectures are on the course website if you’re interested!
- **Topics:**
  - Programming Languages
  - Exceptions in Java
  - Further Advanced Java Topics
Why Research?

- Often different from coursework, opens up the black box
- Get to know professors and undergraduates, M.Sc. and Ph.D. students better
- Lots of interdisciplinary opportunities
- Path to Graduate School
- Be independent and creative
- Learn how to read the latest research literature
- FUN!

What can you do with research?

- Get into Grad School! Get awesome jobs!
- On the path to becoming an Academic
- Work on cutting-edge problems in industry
- Microsoft Research (est. in 1991) is one of the largest, most respected software research organizations
  - They work in a variety of areas like systems, security, data mining and big data, computer graphics and computer vision, speech and natural language understanding, UI and UX, algorithms
  - Typically need a Ph.D. to work there, though some researchers do have developers working with them
  - GoogleX, IBM Research, Adobe Research...

How to Get Involved with Research (1/2)

- Ask specific professors about projects
  - The best way to find interesting options is by talking to professors
  - Be prepared to talk about your interests and background

- Take a 1000-level course
  - Taking upper-level CS courses is a great way to explore your interests and discover what you like

- Take a graduate course
  - Often have a research component to it

How to Get Involved with Research (2/2)

- Many open opportunities for undergrads!
  - Attend the Town Hall meetings every semester
  - Browse cs.brown.edu for info about specific professors and projects
    - Undergrad CS research overview: cs.brown.edu/dept/compsci/undergrad/research
    - Undergrad CS research opportunities: cs.brown.edu/dept/compsci/undergrad/research
  - Meta-URAs (Undergrad Research Assistants) help coordinate research
  - Any undergrad research questions? Contact mura@cs.brown.edu

- Typically start after a few years of CS, but some faculty (e.g. Andy, Stephanie Tellex) take outstanding students after 15/16
Some Brown Research Areas

- Brown Laboratory for Linguistic Information Processing (BLLIP)
- Data Management and Data Science
- Graphics
- Computer Vision and Computational Photography
- Programming Languages
- Theory
- Visualization Research Lab
- Brown Robotics

What “Industry” Looks Like

Industry

Education Tech (“EdTech”)
- Coursera, Khan Academy, Knewton, Canvas
- athenaHealth, HealthCare.gov, Clover
- Robinhood, Wire, Square, Bloomberg
- Google, Facebook, Apple, Dropbox
- Pixar, Dreamworks

Bio Tech
- Ad Tech
- Robotics
- Virtual Reality
- Security
- Databases
- Lots and Lots More...

What “Industry” Looks Like

Industry

- Government
- Non-Profits
- For-Profits

Software Engineer/Developer
- Focus on creating and coding the software
- Variety of specialities: Test/Quality Assurance (QA), etc.
- Not a code monkey, not coding 10-12 hours a day
  - Often in meetings collaborating on design, setting requirements, and talking to prospective customers
  - Depends on company/job, so research/ask about it during process
- Can work on different parts of applications:
  - Specialists: Frontend, Backend, Databases
  - Generalist: “Full-Stack”

Project/Program/Product Manager
- Focus defining what the product should be and what features it should have
- Includes some level of project management/coordinating
- Work with both prospective users and software developers
- Technical position
  - Some PMs code and make prototypes
  - Can’t just tell everyone what to do. Have to convince the engineers that your plans are the best for the product
Types of Careers in Industry (3/3)
- UX (User Experience) Designer
- UI (User Interface) Designer
- Data Scientist
- Systems Programmer
- IT Architect
- And many more!

The Road to a Job/Internship*
- Research companies
- Submit resume (online, at Tech Fair, at Career Fair, by email, etc.)
- Technical interview (phone/skype)
- Onsite Interview
- Offer

*very approximate - many companies will have slightly different steps!

Aside: Technical Interviews (1/3)
- Many software development jobs & internships require some form of technical interviews
- On an online coding pad or on whiteboard, interviewer gives some problem that you have to solve
- In 45 minutes to an hour, you are expected to reason through different ideas and write code or pseudocode for a solution
  - Expect to talk aloud and show your work
  - Your solution doesn’t necessarily have to work or be totally right!
  - Most importantly, they just want to see how you think

Aside: Technical Interviews (2/3)
- If coding, can normally use whatever language you are most comfortable in
- Problems are often algorithmic:
  - “How could you reverse a LinkedList?”
  - “How could you build a Queue using two Stacks?”
  - “Imagine a long scenario about some hypothetical game. How would you account for a specific case or rule?”
- Many of the foundational data structures and algorithms (and methods of problem-solving) needed for technical interviews are covered in lots of depth in CS16 :)
What an Internship Might Look Like (2/2)

- Working at a mid-level - large company (> 200 people)
  - Building many products, can get exposed to multiple different technologies in one company
  - Larger user base → take less risks
    - Might not have as much freedom, but affect many more users
  - Slower paced → features pushed out every couple of weeks/month

Personal Experience:
- Worked as full stack software engineer for Microsoft IoT
  - Had 0 experience with IoT
- Company size: many
  - Larger team → 10-12 people
  - Clearly defined specs and goals for each week/month
  - Lots of collaboration with other teams at Microsoft (less use of open source material)

Department & Culture

Department & Culture

Future Classes

- Some options for next semester
  - CS16: Algorithms and Data Structures
  - CS22: Discrete Math and Probability
  - CS180: Cybersecurity and International Relations
- After that
  - Graphics
  - Systems
  - AI
  - Theory
- No single Brown CS class is required!
- Collaboration in future CS classes!

Involvement at Brown (1/3)

- WiCS - Women in Computer Science
  - Mentorship program, meetings and events
  - Supports the Artemis Project, a free summer camp for rising ninth-grade girls from the Providence area who show interest in science and technology.
- Mosaic+
  - New student group last year, advocating for diversity within Brown's CS community
  - Big-little system, workshops, group study

Involvement at Brown (2/3)

- Hack@Brown
  - Hackathon and year-round workshops, started and run by a team of undergrads
    - Run workshops and events year-round
  - Registration for the hackathon will open soon!
    - You can also apply to be a part of the management team in the fall

Involvement at Brown (3/3)

- The UTA Program
  - 60% of Brown CS concentrators TA at least one semester
  - Applicants open in October for the next Spring semester and March for the next Fall semester
  - TAing is a great way to help students, reinforce CS15 material for yourself, make a little money, and to meet other people in the department!
  - Many past TAs and Head TAs have said TAing was one of the most important parts of their Brown education
Myth: You have to eat/sleep/breathe Computer Science

My experience:
-● Brown CS is incredibly friendly to studying abroad
  ○ Take plenty of courses in other departments
  ○ Lots of people double-concentrate
-● Don’t have to pull all nighters to be a “real” CS student
  ○ Note: There is no such thing as a “real” CS student

Myth: An [ ScB / AB ] is the better option.

My experience:
-● Really doesn’t matter
  ○ Take classes that you find interesting!
  ○ Could do the AB and end up taking extra classes above the minimum

Myth: As a computer science student, you should do a software engineering internship after freshman, sophomore, and junior years to be on the “right track”

My experience:
-● Can take a few summers to explore your interests
  ○ Totally possible to not intern freshman or sophomore year and build your resume in other ways: Research/TAing/other involvement/summer classes/non-CS jobs

Myth: If you’re in a computer science class, you probably want to work in the tech industry

My experience:
-● It’s just a field of study - people come to CS from many backgrounds and graduate CS to do a variety of things
  ○ Some options: engineering, product management, nonprofits, research/academia, completely unrelated work

In Conclusion...
-● Good luck with your final project and the rest of your semester
  ○ You’ve got this!
  ○ Reflect on how much you’ve learned - go back and look at AndyBot, we promise it will be an enlightening experience

-● We <3 Brown CS
  ○ The people in this department are awesome
  ○ This field is awesome
  ○ Computers are awesome
  ○ You are awesome!!!

Announcements
-● Final Project deadlines
  ○ On-time 12/18, 11:59pm
  ○ Late 12/20, 5pm

-● Thank you all for an awesome semester- good luck finishing up the semester!

-● If you see avd on Thursday, say happy birthday!