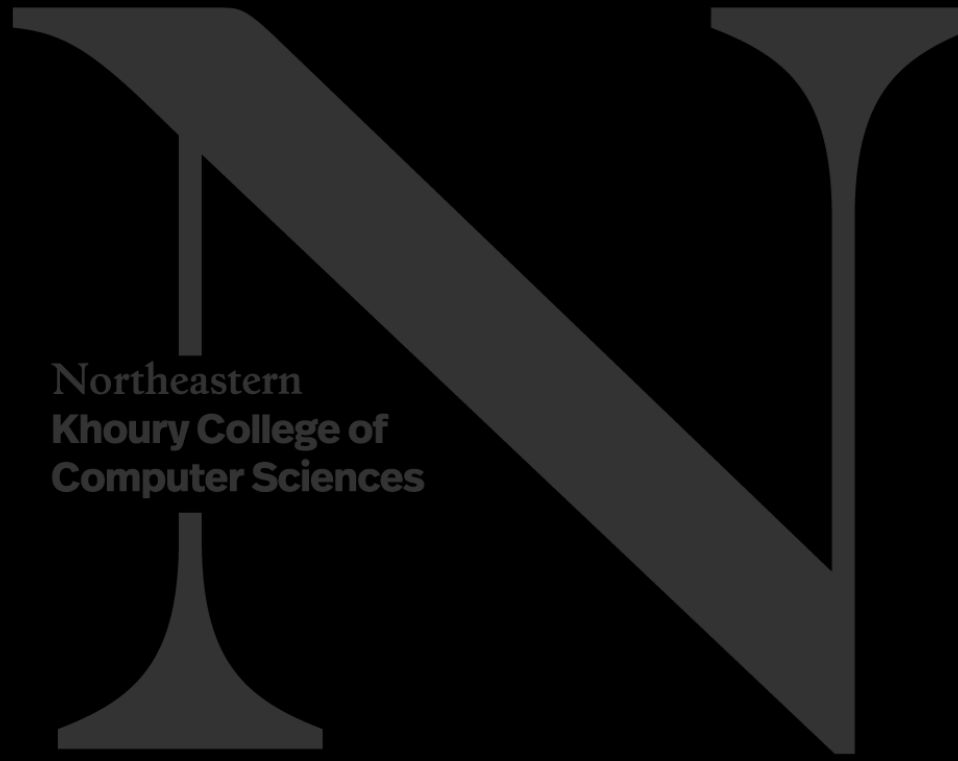
# Welcome - 5004

## Quick Introduction to 5004



# Welcome

This course is **programming intensive!**

- Object Oriented **Design**
  - Object Oriented Programming (java)
  - Process of designing programs
  - Tools to help better design
  - Working with software engineering tools and processes
    - Including a team project
- What it is not:
  - Learn to program java
  - A full software engineering agile/scrum development course
    - I talk about this as part of the starting process – but this is a later graduate course
  - Design patterns
    - Once again, we give samples, but not the full heart of it

# Logistics

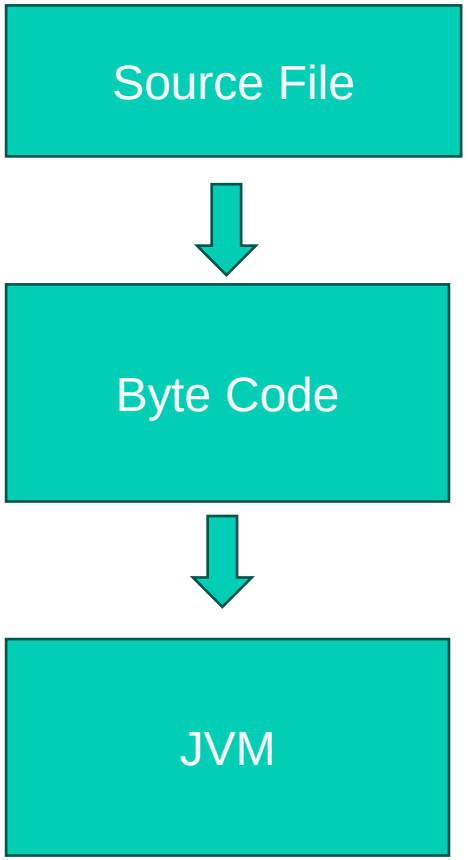
- Assignments
  - Every other week
  - But takes two weeks of programming!!
  - Both a programming and theory section (Report.md)
  - Building up to MSCS course that are more project focused
- Team Activities
  - Assume **two hours** each week!
  - Will assign groups by Monday
- Knowledge Checks
  - Treat them as study guides
  - Randomized questions
  - MUST be done (3/4) to open-up modules.
- Videos and Modules
  - 1 (numbered) a week
  - Notice, not everything has view (module 04… brutal!)
  - I don’t like the videos (not bad, just not how I explain things)
    - Live session will be two hours.
    - First hour or so, answering questions based on teams list
    - Second hour – preview of *next* week. Only a preview – will still need other resources

# Now let's learn java…

- Wait, but you said
  - Languages are tools – we pick the best for the job
- Java's strengths
  - Built around the idea of objects (sometimes to a fault)
  - Strong SDK with rich classes to use
    - Collections, Streams, Networking
  - Write once, run everywhere
  - Extremely popular in industry
- Java's Faults
  - Built around the idea of objects, almost true
  - Older language than python
  - "Classpath" issues not as bad as python, but still not great
  - Attempted to match other languages – too much
    - Java is NOT C/C++
    - Java is NOT javascript! (actually javascript was named after java…)

# Java Environment

- Compiled Language
  - Byte Code
  - Advantage's
    - Type checking at compile time
    - Can optimize code for computers easier
- Java Virtual Machine
  - Runs optimized byte code on your computer

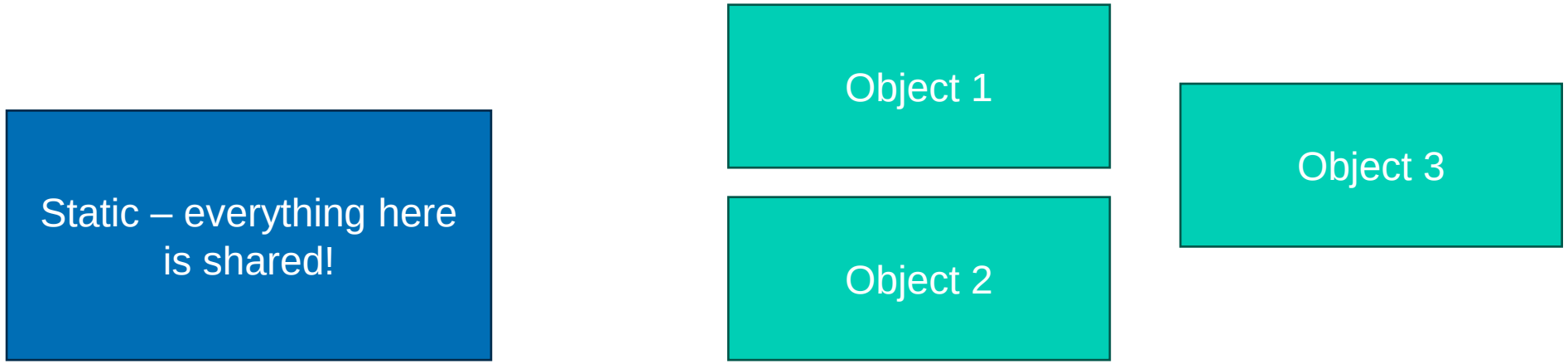


# Primitive vs. Objects

- Confusing concept? (ok may be others)
  - Primitives
    - These are “raw” data types (numbers)
      - int, short, long, float, double, boolean, char
    - They don’t have functionality (methods)
      - Integer, Short, Long, Float, Double, Boolean, Character == matching objects with functionality
    - Can be stored in arrays (blocks of allocated memory)
  - Objects
    - Has data (other objects or primitives)
    - Has functionality via methods
    - Can be stored in arrays or any other object (Collections framework)

# Another confusing concept

- Static vs. Non-static
- Memory allocation idea (covered more in 5008)
- Static == shared memory
- Member (non-static)



# So… objects review

- Everything in java is a “Class”
  - Think of it as a blueprint
  - Contains both static and member/non-static information
- Covered objects at the end of 5001
- Java is structured around this idea
  - No writing functions without a blueprint around them!
  - Can be confusing at times, as sometimes you just need a function
    - If you make a method “static” is turns into a function essentially

# Let’s work through code to see examples

And then onto homework 01! (we will complete most of it this afternoon)