David M. Keil, Framingham State University

CSCI 252 Computer Science II Using Java

Introduction

1. What this course offers
2. Course topics
3. How the course delivers

Self-introductions

- I teach programming and theory courses
- My research is in concurrent computing
- Who is here?
- What were your experiences in CSCI 130?
- What do you expect from this course?
- How confident do you feel with strings; loops; debugging; methods; files?
- How do you learn?

They made programmable computers possible

Ada Lovelace *wrote the first computer program*

John von Neumann *first developed the idea of storing programs as data*

They made Java possible

Noam Chomsky *described linguistics concepts that are used in programming languages*

Grace Hopper *invented the COBOL language*

James Gosling *developed Java, using the virtual-machine concept*

1. What this course offers

Inquiry: What will *you* need to know about computer science and *intermediate* Java programming, five years from now?

- as an IT professional
- as a software engineer
- as a CS major

- *Case:* How would *you* build computing tools to help construct a map of the brain?

What CS II is about

- *Software development* and *modular design* of larger programs
- *Object-oriented design and programming:* encapsulation, containment, inheritance, polymorphism
- *Other issues:* nested loops, time performance, recursion, exception handling, graphics, GUIs, multi threading

The chronic software crisis

- The global market demands that new tools be created to work and do business
- New software is required rapidly
- Software development has *always* tended to be behind schedule
- Software is often unreliable
- *Solutions:* web hosting, design, documentation, readable code, structured techniques, object-oriented technology

Cases

- Hacking that exploits security faults
- Mass data collection by government, corporations
- Disasters with medical, aeronautic, voting-machine software
- *Challenges:*
  - Writing secure software
  - Concurrency, threading, robotics

CS II offers an environment to

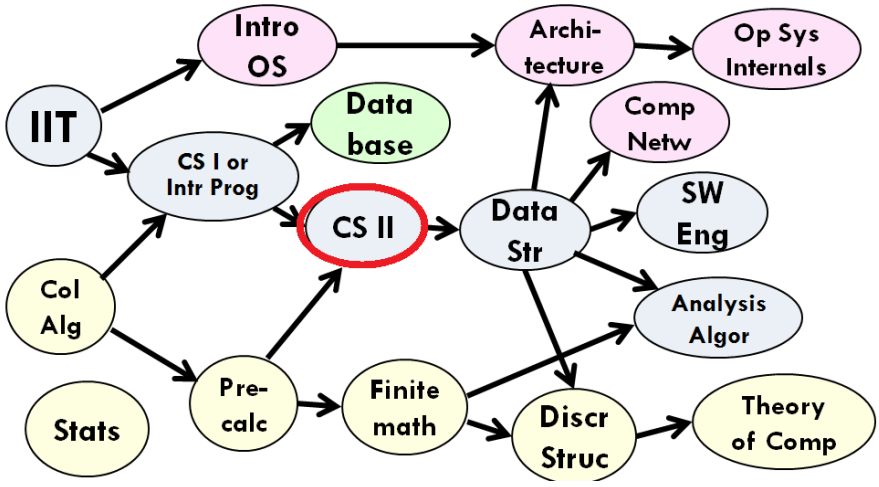
- Build your problem-solving capabilities
- Investigate computing concepts
- Expand skills with Java
- Work on a team
- Discuss what it takes to create better software

The CS curriculum

Threads:

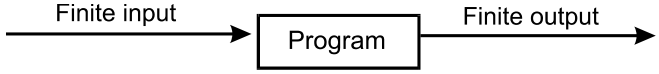
- **Programming**, system design, and software engineering
- **Computer architecture**, operating systems, and networks
- **Data management** (e.g., databases)
- **Theory** (logic, statistics, discrete math, algorithm analysis, automata, AI)

How CS II fits into the CS-G concentration

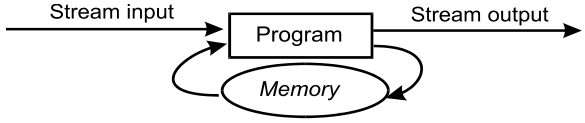


Three computing paradigms

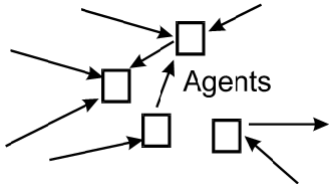
- *Algorithmic (T1-3):*



- *Sequential-interactive (T4-6):*



- *Multi-stream interactive (T7):*



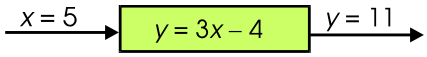
2. Course topics

Introduction and background

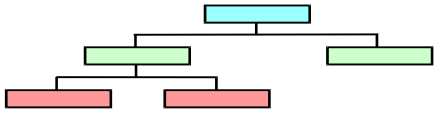
1. Method design
2. Arrays and loop design
3. Class design
4. Collections
5. Inheritance and polymorphism
6. Event-driven GUIs and graphics
7. Concurrency and mobile and web apps

Some CS II topics in diagrams

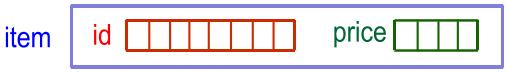
- Method



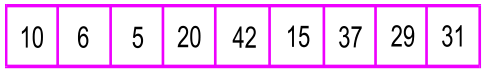
- Module hierarchy



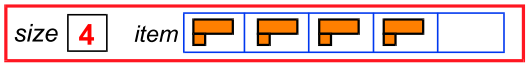
- Object



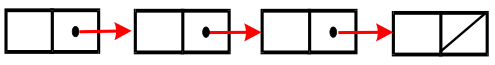
- Array or random-access file



- Collection



- Linked list



CS II course-wide objectives

a. Participate in class activities throughout the semester
b. Solve problems as part of a team
c. Work at the blackboard
d. Present written results
e. Show knowledge of facts and concepts
f. Summarize the semester's learning
g. Complete a software project

Topic 1-4 objectives

1. Define and test Java static methods with parameters and return values.
2. Define and safely manipulate arrays, designing nested loops and applying search and sorting algorithms.
3. Define and test Java classes, explaining object-oriented design concepts.
4. Design, implement, and explain a multi-class application that manages a disk-based collection.

Topic 5-7 objectives

5. Explain and implement the notions of an inheritance hierarchy and of polymorphic behavior.
6. Explain event-driven GUI development and use Java graphics libraries.
7. Describe concurrent features of mobile and web environments, and implications for programming, ethics, and security.

Essential subtopic objectives

0.3a Trace a branch or loop
0.3b Solve a numeric loop problem
0.4b Describe memory allocation for objects
1.1a Explain procedural abstraction
1.2b Write a method with parameters and return values
2.1b Describe Java arrays
2.2a Traverse an array
2.4a Give the output of a nested loop
2.4b Write a nested loop
3.1a Describe Java data abstraction
3.1b Define a Java class
4.1 Explain what a collection is

Essential subtopic objectives (programming)

0.3c Solve a loop problem with strings

0.3d Debug a defective loop

1.1b Define and test a Java method

1.5b Read a file using a loop

3.3b Test and debug a class

Objectives assessed by multiple-choice quizzes

Recall:

0.0a basic Java/programming concepts*

0.0b loop concepts*

0.0c bitwise-operator concepts

1.0 basic Java method concepts*

2.0a basic array concepts*

2.0b advanced array concepts

3.0a basic class concepts*

3.0b advanced class concepts

4.0a basic collection concepts*

4.0b advanced collection concepts

5.0a basic inheritance concepts*

5.0b advanced inheritance concepts

6.0a basic graphics concepts*

6.0b advanced graphics concepts

7.0 concurrency concepts*

Breakdown of subtopic objectives

	Essential		Priority		Challenge	
Topic	Total	Exer-cises	Total	Exer-cises	Total	Exer-cises
Backg	5	2	12		2	
1	4	2	4		2	1
2	4		3	2	5	2
3	3	1	2		4	2
4	1		4	2	7	3
5			4	1	6	2
6			3		5	3
7			2		7	3
	17	5	34	5	38	16

Notes:

- Essential objectives matter most
- Early topics have many objectives that matter; later topics have few

Some themes of CS II

- Defining a *class* lets us model the objects found in a problem domain
- A *collection* class lets us model a group of similar objects
- We implement a collection using an *array* of objects, or a *linked* structure
- *Algorithm analysis* helps manage time complexity of processes
- *Software engineering* helps manage complexity of developing a solution

Features of programs we will work with

- Longer than in CSCI 130 or 152
- Program size forces modularity
- We may need to focus on one part of a program at a time
- *Algorithm choice*, *procedural abstraction*, and *data abstraction* are part of the design

Sequential and random access

Stream and list data is *sequential*

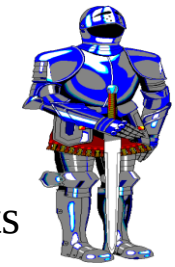
- Access is from start to end
- *Examples:* keyboard (*Scanner*), screen (*System.out*), text file

An alternative is *random access*

- Program may store or retrieve an arbitrary item of a collection
- *Examples:*
  array; ArrayList; random-access file

Software quality issues

- *User interface* must be clear and straightforward
- *Robust* ("bullet-proof") software can handle any error condition
- Program should give correct results (including error messages if necessary) on *any* input
- Ensuring quality begins at the specification stage



Kinds of errors

- *Syntax* (program won't compile)
  - We will formalize Java syntax
  - Compiler error messages may mislead
- *Logic* (output is wrong)
  - What *intermediate value* is wrong?
  - *Tracing* is the basic tool
  - Software engineers use deduction like a detective
- *Runtime* (handle with exceptions)

3. How this course will deliver

- How do you expect a course to be organized?
- How do you know what you're learning?

This course is an inquiry

- *FSU motto:* "Live to the truth"
- What does this mean?
- What is the truth?
- What does *truth* mean in software development and computer science?
- *Possible ways to the truth:*
  - Authority of experts
  - Our own experience, e.g., testing
  - Discussion

Classroom format

- Emphasis is on *inquiry*, *activity*, and *collaboration*
- *Slides and short presentations* summarize the content of the course
- We seek to create a *natural critical learning environment*
- *Your contribution and participation* matter

In this classroom

- We work on different problems
- We scratch our heads
- Brilliance means getting it after effort
- Everyone is brilliant sometimes
- Mistakes are intriguing

Course organization

- This course has seven *topics*; each with an *objective* and 3-5 *subtopics*
- Each subtopic has one or more *objectives*
- I label subtopic objectives *essential; priority;* or *challenge*
- A set of *problems* for each subtopic objective is available
- Exercises and quizzes focus on these

Exercises

- *Exercises* consist of individual and group problem solving
- Each student solves problems from as many essential and priority outcomes as possible; I check off work received
- Before solving a problem on a quiz, solve one for that objective as an exercise
- Some exercises I evaluate for achievement of objectives

Standards for programming exercises and projects

- Comment your code; include your name, the date, and the topic and assignment number
- All methods and loops need comments giving purpose
- Provide adequate tests of your code

Due dates

- Please bring exercises on the last day we discuss a topic; we'll have a quiz then too
- Exercises more than 2 weeks late count 50%
- Week 6 is deadline for mid-semester assessment of all work
- Preliminary version of project is due Oct. 22, full version Dec. 3
- I'll give feedback so students may fill gaps and make corrections

Six-week tally

- Please master *foundation material* for the course (topics Intro to 2) by mid-semester
- Your ability to *contribute* and to *learn* will depend on that; so will success on topics 3-7
- Mid-semester assessment:
  - Scores for objectives
  - Exercises and project work
  - Solutions presented in class
  - Attendance and preparation

Role of groups

- Assign objectives to group members for presentations
- Help each other with exercises
- Collaborate in classroom exercises
- Collaborate on part 2 of semester project

Group work in classroom

- Form groups of 3 students to take up a problem
- Each group should have
  - *Facilitator* – keeps discussion on track
  - *Recorder* – writes results of discussion
  - *Reporter* – presents results to class
- Participation by all in group work is one of our basic objectives in this course

Attendance, participation, and preparation

- Please prepare for class discussion; e.g., by solving an exercise problem
- Bring evidence of preparation to each class
- Some students in each class session will show problem solutions

Assessing objectives in class

- After doing exercises on a topic, a student may show attainment of an outcome/objective by solving a quiz problem, in writing, in class
- More opportunities will be available for each outcome
- The main factor in success is attaining objectives and outcomes
- Another factor: contribution to everyone's learning

Scoring answers, not quizzes

- A wrong answer means "try again"
- Scores on outcomes can only improve as you learn more
- Leaving a question blank or "I don't know" often shows discernment
- Worst possible result is "Not yet"

What is a successful solution?

- Reflects understanding of
  - question or problem
  - relevant concepts
  - relevant procedures
- Usually reflects study and work on an exercise

Pretest

- This pre-quiz will assess some capabilities presented in CSCI 130
- Many of these are considered essential CS II objectives
- There will be more chances to show success with the objectives assessed

summary quiz and final-exam day

- During the last week of classes, we'll have a *summary quiz* of problems and multiple-choice questions
- On final exam day, students will talk about their semester projects, and we'll have make-ups on summary quiz

Programming project

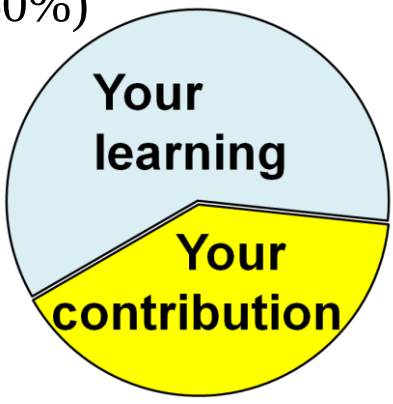
- As a *semester project*, you will construct two *applications* in Java
  1. completion of a basic file-maintenance project: input, display, store, and retrieve data about items of your choice
  2. your choice among a set of Java programs with multiple classes
- The project goal is experience in specification, design, coding, and testing

Six-week tally

- Please master *foundation material* for the course (topics Intro to 2) by mid-semester
- Your ability to *contribute* and to *learn* will depend on that
- So will success on topics 3 to 7
- Mid-semester assessment:
  - Scores for objectives and quizzes
  - Exercises and project work
  - Solutions presented

Assessment and grading

- *We measure:*
  - Individual achievement of learning objectives (60%)
  - Contribution to the learning of the class (40%)
- *Assumptions:* Learning is *active*, *shared*, and *measurable*



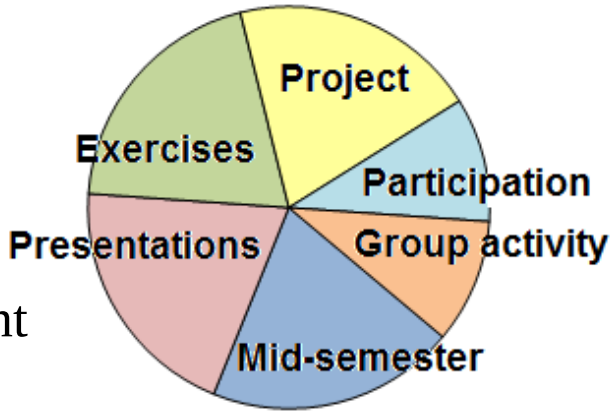
Assessment of learning

Assumptions:

- We can measure learning via objectives
- Some concepts matter more than others
- We learn by *summarizing* and *reflecting*

Assessment of contribution

Assumption:
We learn by:



- Sharing inquiry
- Being present
- Preparing
- Solving problems together
- Activity throughout the semester

Grading weights

Essential objectives	30 %
Priority objectives	20
Challenge objectives	10
Summary quiz	10
Exercises	10
Project	10
Mid-semester tally of work	10
Presentations and blackboard work	10
Documented group work	5
Attendance and preparation	5

The growth mindset

- *Researchers report:*
  - People can learn new skills when they believe that their effort matters
  - Learning takes effort
  - Intelligence can grow with effort
- Alternative mindset: *fixed*
- The fixed mindset says that innate talent, not effort, is decisive and changeless

Hidden curiosity and talent

- You were born curious; it's in your nature
- Schooling can suppress curiosity
- One option is to allow our curiosity to re-emerge as part of our true selves
- For me, this can enable "effortless" effort and helps me to be more present with what I study

Quality and learning

- People enjoy doing quality work
- It requires freedom of choice and control of the work environment
- Part of learning is recognizing quality work
- *You*, *other students*, and *I* can help you to see quality of your work
- Research predicts that if we know quality, we will produce it

Grades, learning, and effort

- Learning requires *curiosity*, *intention to learn*, and *undistracted effort*
- Attention to grades distracts from what we're learning
- If grades measure learning, then:
  - *Getting higher grades requires paying less attention to grades!*

Academic integrity

- Directly lifted text must be quoted and credited
- Use of ideas or other information must be credited by citations or references
- Citation standards for MLA and APA are given at *www.citationmachine.net*
- *Plagiarism:* "occurs when you use someone else's ideas or words and represent them as your own." Cite your friends!
- See catalog for FSU policy

What signing work means

- In this course, all code and words submitted are to be of the *student who signs the work*
  - *Quizzes*: no collaboration or device use
  - *Exercises:* device use and collaboration are recommended
- *Principles:*
  - *Words* belong to the original writer
  - *Ideas* belong to everyone; but we acknowledge their sources

A proposed agreement

I commit to:

- know the course material, present it clearly
- return submitted work within a week.
- welcome questions and answer them helpfully

You commit to:

- prepare for class and submit evidence of it
- ask questions
- answer reasonable questions, risking error
- work sometimes in groups
- present results or lead discussions

References

C. Dweck, *Mindset.* Ballantine, 2006.

W. Glasser, *The Quality School.* Harper, 1990.