Introduction to CS102

- Objectives
  - Undertake real-world design task
  - Work as a member of a team
  - Practice communication in written & oral form
  - Learn more programming techniques
  - Practice independent learning!
- General
  - Transform basic computer literacy, design and programming skills you learnt in CS101 into practice
  - Expand the range of techniques you have to solve problems

This course should help you…

- improve your programming abilities
  - Enhanced OOP
  - GUI & Event-driven programming
  - Recursion
  - Data structures

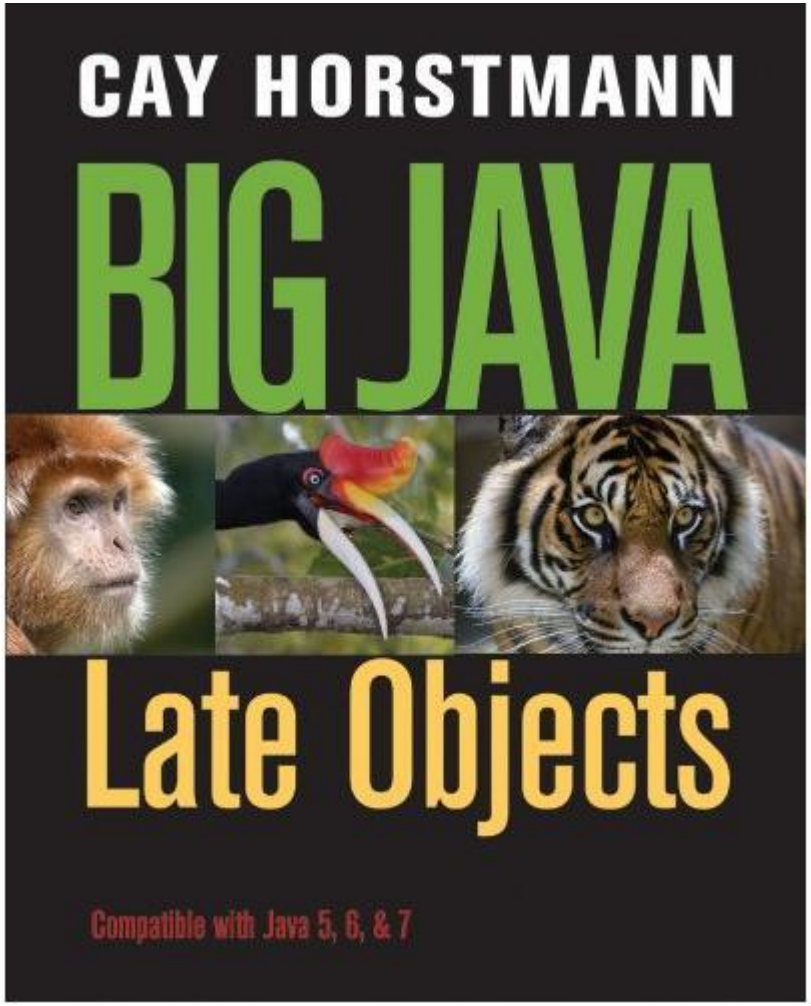
- practice core engineering skills
  - Written & oral communication
  - Teamwork
  - Independent learning

Components of the course

- Topics to be covered
  - Recursion
  - Files
  - Some basic data structures
  - Object-oriented programming
  - Event-driven architectures
  - Searching and sorting
- Project
  - Commercial-quality program
  - Fully documented
  - Bug-free and easy to use
  - Group project
  - Written reports and presentations (requirements, specifications, detailed design, user manuals)

Textbook

- Textbook: Big Java (Late Objects) Enhanced eText 2nd edition., Cay S. Horstmann, 2016, Wiley



Course Rules

- Grading (Tentative):
  - 15% – Lab assignments (& contributions to course forum)
  - 25% – Midterm Exam
  - 25% – Final Exam *(see note below regarding examinations)*
  - 15% – * Reports, Presentations & Participation
    { Requirements & User Interface 10%, Detailed Design 5% }
  - 10% – * Demonstration, Final code & documentation
    { inc. wiki & peer grade! }
  - 10% – Homework & Quizzes

Minimum course requirements

- More than 30% on the midterm exam
- More than 75% lab average
- Personal project logs properly completed each week
- Reasonable contributions to each stage of the project.

Grading Scales

Labs

(0) incomplete attempt

(20) weak

(80) close to complete, but not fully correct,

(100) complete and correct.

Note: Students are encouraged to correct any mistakes and resubmit assignments (prior to the deadline.) The objective is to learn!

Projects

(10) excellent almost impossible!

(8) good

(6) ok but could be better

(4) weak definitely not up to scratch, more effort needed.

(0) no real attempt!

Course – Misc.

- Labs are due in week 4
- Moodle – check frequently!
  - Register
  - Schedule
- See also (your section's webpage)

  http://www.cs.bilkent.edu.tr/~gudukbay/cs102/cs102.html

- Cheating/Plagiarism!

TODO

- Enroll to Moodle
- Lab assignment 1 (due week 4)
- Find group & project
  - Group/project selection stage report is Friday Feb. 18
- Any questions?

What has been covered in CS101

- Introduction
  - Computer processing, hardware components, etc.
- Syntax and Semantics of Java
  - Identifiers, assignments, precedence rules, …
  - Console input/output
  - String class
- Flow of control
  - Conditional statements
    - boolean expressions
    - if/else
    - switch
  - Loops
    - for
    - while
    - do while

What has been covered in CS101

- Classes and Objects
  - Instance variables and methods
  - Public/private variables
  - Constructors, mutators, accessors, copy constructors
  - Static methods and variables
  - Memory and references (deep/shallow copy)
- Arrays
  - Creating, accessing
  - Length
  - Bounds of array
  - Multidimensional arrays
  - Ragged arrays
- Enumerated types
- ArrayList/Generics

Syllabus

- Writing classes
  - Objects
  - UML Diagrams
  - Encapsulation
  - Determining the classes and objects that are needed for a program
  - Relationships that can exist among classes
  - Interfaces
  - Method design and method overloading
  - Inheritance, multiple inheritance
  - Class hierarchies
  - Overloading vs. overriding
  - Protected/super
  - Polymorphism

Syllabus

- Sorting
  - Selection
  - Insertion
  - Other sort algorithms
- Search
  - Linear
  - Binary
- Exceptions
  - Exception handling
  - Exception propagation
  - finally/throw statements
- Recursion
  - Recursive thinking
  - Recursive programming
  - Indirect recursion
- Introduction to abstract data types
  - Queues
  - Stacks
  - Trees

Syllabus

- GUI
  - How to use graphical user interface components
  - GUI layouts
  - Polygons and polylines
  - Events: mouse, keyboard…
  - Event adapter classes
  - Timer class
  - Sliders
  - Split panes
  - Scroll panes
  - Combo boxes
  - Recursion in graphics

Program Development

- The mechanics of developing a program include several activities
  - writing the program in a specific programming language (such as Java)
  - translating the program into a form that the computer can execute
  - investigating and fixing various types of errors that can occur
- Software tools can be used to help with all parts of this process

Language Levels

- There are three programming language levels:
  - machine language
  - assembly language
  - high-level language
- Each type of CPU has its own specific *machine language*
- The other levels were created to make it easier for a human being to read and write programs

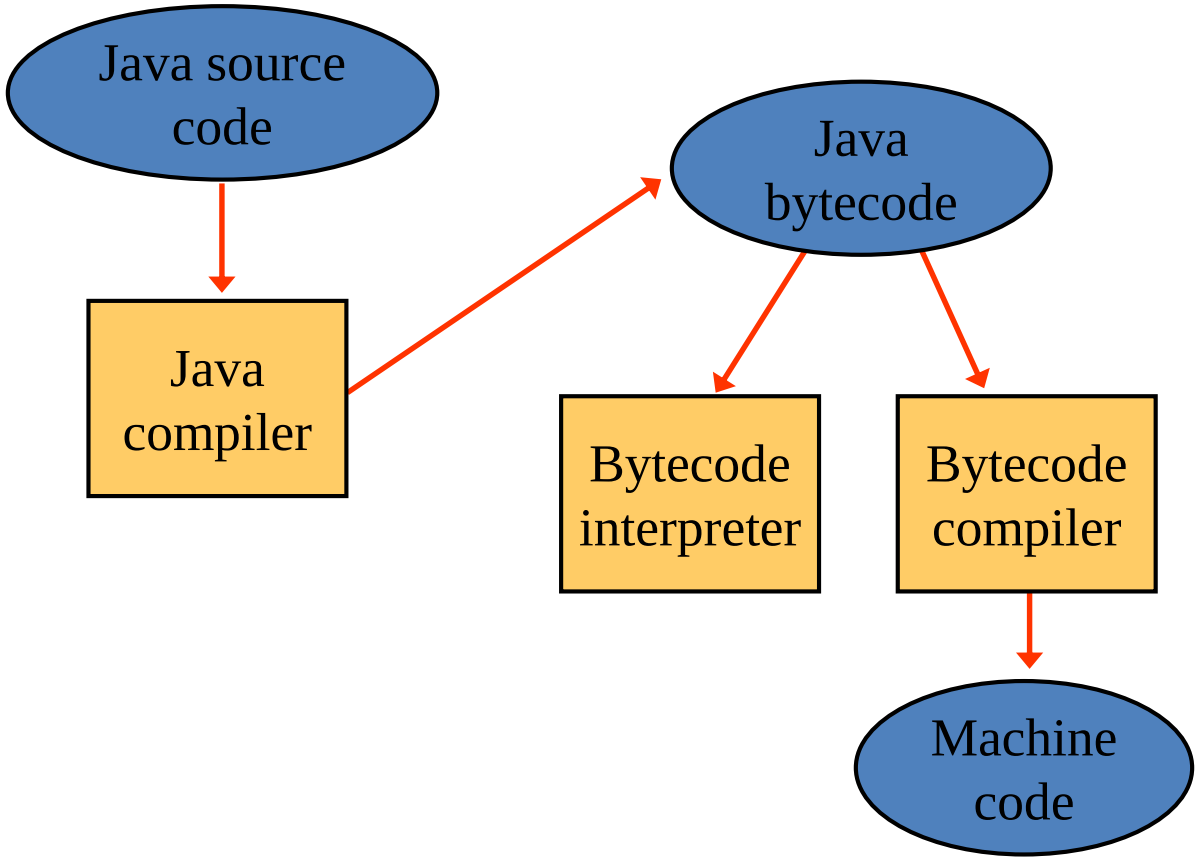
Programming Languages

- Each type of CPU executes only a particular *machine language*
- A program must be translated into machine language before it can be executed
- A *compiler* is a software tool which translates *source code* into a specific target language
- Often, that target language is the machine language for a particular CPU type
- The Java approach is somewhat different

Java Translation

- The Java compiler translates Java source code into a special representation called *bytecode*
- Java bytecode is not the machine language for any traditional CPU
- Another software tool, called an *interpreter*, translates bytecode into machine language and executes it
- Therefore the Java compiler is not tied to any particular machine
- Java is considered to be *architecture-neutral*

Java Translation



Development Environments

- There are many programs that support the development of Java software, including:
  - Sun Java Development Kit (JDK)
  - Sun NetBeans
  - IBM Eclipse
  - Borland JBuilder
  - MetroWerks CodeWarrior
  - BlueJ
  - jGRASP
  - Visual Studio Code
- Though the details of these environments differ, the basic compilation and execution process is essentially the same

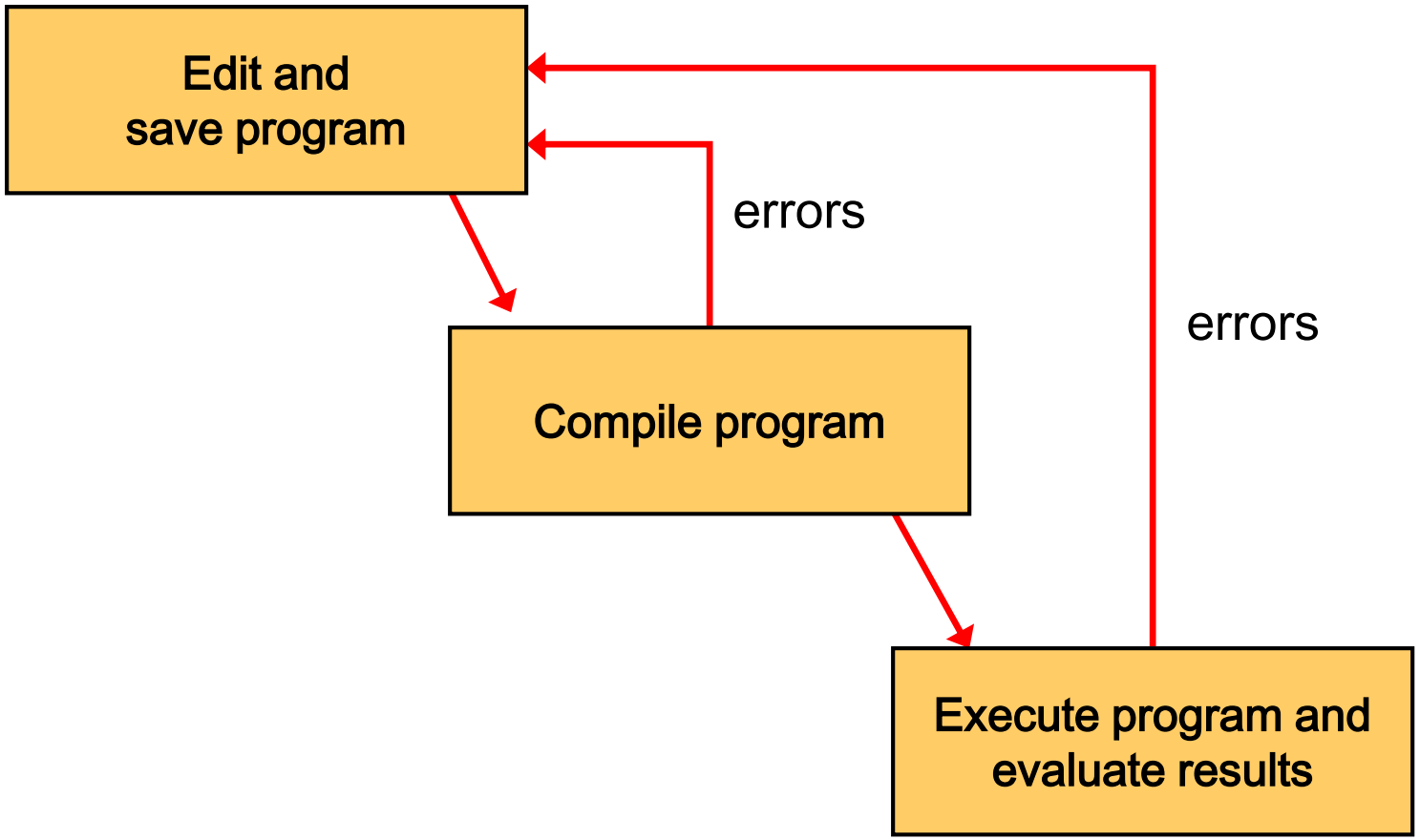
Syntax and Semantics

- The *syntax rules* of a language define how we can put together symbols, reserved words, and identifiers to make a valid program
- The *semantics* of a program statement define what that statement means (its purpose or role in a program)
- A program that is syntactically correct is not necessarily logically (semantically) correct
- A program will always do what we tell it to do, not what we meant to tell it to do

Errors

- A program can have three types of errors
- The compiler will find syntax errors and other basic problems (*compile-time errors*)
  - If compile-time errors exist, an executable version of the program is not created
- A problem can occur during program execution, such as trying to divide by zero, which causes a program to terminate abnormally (*run-time errors*)
- A program may run, but produce incorrect results, perhaps using an incorrect formula (*logical errors*)

Basic Program Development



Problem Solving

- The purpose of writing a program is to solve a problem
- Solving a problem consists of multiple activities:
  - Understand the problem
  - Design a solution
  - Consider alternatives and refine the solution
  - Implement the solution
  - Test the solution
- These activities are not purely linear – they overlap and interact

Problem Solving

- The key to designing a solution is breaking it down into manageable pieces
- When writing software, we design separate pieces that are responsible for certain parts of the solution
- An *object-oriented approach* lends itself to this kind of solution decomposition
- We will dissect our solutions into pieces called objects and classes

Object-Oriented Programming

- Java is an object-oriented programming language
- As the term implies, an object is a fundamental entity in a Java program
- Objects can be used effectively to represent real-world entities
- For instance, an object might represent a particular employee in a company
- Each employee object handles the processing and data management related to that employee

Objects

- An object has:
  - *state* – descriptive characteristics
  - *behaviors* – what it can do (or what can be done to it)
- The state of a bank account includes its account number and its current balance
- The behaviors associated with a bank account include the ability to make deposits and withdrawals
- Note that the behavior of an object might change its state

Classes

- An object is defined by a *class*
- A class is the blueprint of an object
- The class uses methods to define the behaviors of the object
- The class that contains the main method of a Java program represents the entire program
- A class represents a concept, and an object represents the embodiment of that concept
- Multiple objects can be created from the same class

Objects and Classes