## I. Object Oriented Concepts                    week 2

- see web page definitions

## II. Introduction to Java Language Program Elements

- languages can be described as verbose or terse, depending upon the number of **keywords** they contain
- keywords are predefined reserved identifiers that have special meanings to the compiler
- Java # is a relatively terse language, see list of keywords: google Java keywords
- NOTE: keywords are case sensitive!!!!!!!!!!!!!!!!!!!!

### i. comments

- **why?**
  - ➢ documentation for you, or more importantly, someone else in the future
  - ➢ debugging statements
  - ➢ removing sections of potentially poor/damaging code
- single-line comments (also called in-line)
  - ➢ uses the // notation
  - ➢ doesn't need an end symbol, ended by EOL/CR
  - ➢ can go anywhere on line, but everything following is a comment
- block comments (also called multi-line)
  - ➢ uses the /*      */ notation
  - ➢ must have an end symbol
  - ➢ can be on a single line or span across many
- doc comments
  - ➢ uses the /**      */ notation
  - ➢ used by javadoc to generate documentation

ii. **class definition**

- fundamental building block of Java programs
- every application begins with a class definition

iii. **code block definition { }**

- used to define a block of code, i.e. the beginning and end
- each { must have a matching } or errors will occur

iv. **statement terminators**

- terminate complete program instructions, i.e. statements
- use the ; to terminate statements

v. **main method definition**

- entry point for all Java applications, i.e. where the program will begin execution
- general syntax (or signature): public static void main(String[] args)
- public and static keywords are access modifiers (more later)
- void is the return type of the main () method

vi. **main () body**

- contains the code to be executed when the application begins

## III. Errors

1. compile-time (or syntax) errors

   - when: at compile time, i.e. caught by Java compiler
   - why: violations of Java keyword syntax
   - common syntax errors
     - ✓ case errors
     - ✓ misspellings
     - ✓ forgotten semi-colons
     - ✓ missing closing characters

2. run-time errors

   - when: during program execution
   - why: unhandled errors, e.g. math errors like division by 0, invalid user input, file not found errors

3. logic

   - when: typically, the worst possible time
   - why: complex problems, less than optimal time spent on design

Which one's of these are the easiest to find/fix? Which are the hardest?

How to avoid logic errors?

- good analysis and design techniques
- use of planning techniques, e.g. flowcharts, algorithms

## IV. Java and the NetBeans Development Environment

- Java is a language created by Sun Microsystems starting circa 1991
- Sun Micro was purchased by Oracle in 2009, thus Oracle now owns Java

- the NetBeans development environment called NetBeans **IDE** – Integrated Development Environment and is separate from Java

- java source files should end in .java
- javac is the name of the java compiler, once run, this creates a .class file
- from the console window, java will execute the .class file

- main method of organization called an application
- projects divided into multiple, physical **files**
- **ALWAYS** create a folder per project…save all of your files to this folder, and **ONLY** move the folder as a whole

- for example, create a OOP 1 folder, then using project name & location to create a subfolder with that project name

- on startup, NetBeans displays a "start page", where you can **<u>Open</u>** an <u>existing</u> project or open a **<u>New</u>** project (not file, PROJECT)

- if you select **<u>New,</u>** you will be prompted for:

  - Project category: Java with Ant
  - Projects: Java Application

  - Project Name: name using each word upper case, no spaces, using good names
  - Project Location: if populated, leave alone (e.g. NetBeansProjects)
    otherwise, create folder for projects, then enter entire path name

  - Create main class, use defaults (leave defaults alone)

- to (build and) **<u>RUN</u>** project, click the start button (>) in the toolbar, or click Run…run project in the menu bar, or press SHIFT F6

  (Note, if simple programs will not display output correctly, try SHIFT F11, then run project)