Review

Fall 2015
CMPU 102
Computers think in bits (1 or 0)
- 00101001 = 81

Eight bits per byte
- 1024 bytes = 1 Kb
- 1024 Kb = 1 Mb
- 1024 Mb = 1 Gb
Computer organization

- CPU = brain

- Microprocessor: computer on a single chip

- Network: when two or more computers can communicate directly, they form a network
Software

- Operating System: the program that manages a computer’s resources

- Program: a sequence of instructions that performs some task
  - Performing an instruction is called “executing” an instruction
Compilation

- **Translator:**
  - translates a program from one language to another

- **Machine language:**
  - the ones and zeros that a computer understands

- **Compiler:**
  - a translator which typically translates a high-level language into a low-level one
  - Java is a high-level language

- Java’s compiler translates Java code into **bytecode**
  - Bytecode is like machine language, but not tied to a specific machine

- A Java bytecode interpreter is used to execute the bytecode
  - Called a Java Virtual Machine (JVM)
Terminology

- Abstraction
  - Similar objects exhibit similar behavior
  - The ability to do the same “thing” on many objects, disregarding details

- Encapsulation
  - Not revealing how the method does it’s work

- Modularity
  - Dividing code into smaller pieces (modules), each one of which is easier to code
OOP Terminology

- OOP (Object-Oriented Programming) languages:
  - Abstract actions into the class methods
  - Encapsulate code inside the class methods
  - Use additional methods for modularity

- A (primitive) type is the basic unit of storage in Java
  - A type is a template for a variable

- A class is composed of variables as well as methods
  - A class is a template for an object

- Creating a variable/object from a type/class is called instantiating a class type or initializing a primitive type
Problem solving steps

1. Analysis
   - What needs to be done?

2. Design
   - How is it going to be done?

3. Implementation
   - Make it happen!

4. Testing
   - Does it work correctly? If not, repeat steps 2 – 5.
Readable programs

- **Comments** are English text
  - `//` for a single line comment, all code to right is comment
  - `/* */` multiple line comments
  - `/** */` multiple line javadoc comments

- **Blank lines** create vertical separation and make a program easier to read

- **Indentation** helps humans identify which code is contained within a set of braces

- **Keywords** have special meanings in Java; can’t be used for identifier names
  - Examples: `int`, `double`, `class`, `static`, `public`
Identifiers

- Identifiers: programmer-defined names
  - For classes, variables, methods, etc.
  - Cannot be a keyword
  - Must start with a letter (or _ or $)
  - Can contain numbers also (but not as the first character)

- Good identifiers: radius, width, position

- Bad identifiers: x, y, q,
  the_really_really_long_variable_name_hi_mom
Computer bugs

- A bug is an error in the program
- To debug is to remove bugs
- We learned methods to test code
Java classes

- The `class` keyword is used to start a class declaration
  - Can be made public

- A class can be a library of static methods

- A class can be a “template” for objects
  - Just as a type is a “template” for a variable
Java methods

- All method signatures have the following syntax:

  ```
  modifiers type name ( parameters ) { statements }
  ```

  - **Properties of the method**:
    - public
    - static
    - void
  - **Type that it returns**: main
  - **A name for the method**: main
  - **Any number (including zero) of parameters**: String[] args
  - **The body of the method (can be empty)**: ```{ ... }```
Program execution

- Java starts executing a program at the beginning of the main() method

- Braces { } are used to specify where a method or other body of code begins and ends

- A **statement** ends when a semicolon is encountered
  - A statement can span multiple lines
A literal character string is a sequence of characters enclosed by double quotes (e.g. “Hello world!”)

System is the Java class that allows you to access parts of the computer system
- System.in: access to the keyboard
- System.out: access to the monitor

Period is used for selection: Math.round
- Given String s, select a method via: s.substring()

An exception is when Java “panics”
- It means something is wrong
Java provides escape sequences for printing special characters. These must be used inside quotation marks "".

- \b backspace
- \n newline
- \t tab
- \r carriage return
- \\ backslash
- \" double quote
- \' single quote
Java has 8 primitive types:
- float
- double
- boolean two values: true and false
- char a single character (inside ‘’s)
- byte
- short
- int
- long

The 4 bolded primitive types are the most commonly used.
Constants vs. literal values

- Which is easier to enter:
  - Math.PI
  - 3.141592653589793

- Entering a symbolic name (i.e. a constant) reduces chances of errors

- Constants are always final
References and variables

- A **variable** is an actual spot in memory that holds a (primitive type) value.

- A **reference** is a memory address that points to another spot in memory where the object is actually stored.

- Variables defined in a class but not inside a method are initialized to a default value (global variables).

- Variables defined in a method are **not** initialized to a default value (local variables).
Math

- Standard operators: + - * / (using infix notation)

- Note that / can be either integer division or floating-point division

- % computes the remainder (aka modulus)
Expressions

- Evaluating an expression yields a result and a type
  - Example: 4/3 yields 1 of type int
  - Example: 3.5*2.0 yields 7.0 of type double

- Binary operators have two operands
  - Example: 3+4, 6*3, etc.
  - Left side is evaluated first

- Unary operator has one operand
  - Example: -3, etc.

- Operators have precedence so parens are needed at times
  - For example, * and / are evaluated before + and -
Operators

- Assignment: =
- Increment (++) and decrement (--) unary operators
- Consider:
  ```java
  int i = 5;
  System.out.println (i++);
  System.out.println (i);
  System.out.println (++i);
  System.out.println (i);
  ```
- There are 4 ways to add 1 to an int:
  ```java
  i = i + 1;
  i += 1;
  i++;
  ++i
  ```

There are many such compound operators
Casting

- Casting converts one type to another

Example:

```java
int x = 1;
System.out.println ((double) x); \rightarrow 1.0

double d = 3.4;
System.out.println ((int) d); \rightarrow 3
```
Creating one:
Scanner stdin = new Scanner (System.in)

Methods in Scanner class:
- public int nextInt()
- public short nextShort()
- public long nextLong()
- public double nextDouble()
- public float nextFloat()
- public String next()
- public String nextLine()
- public boolean hasNext()
References

- An object variable is really a reference to an object
- null represents an object variable that points to nothing
- Once nothing points to an object, Java automatically deletes that object
  - Called **garbage collection**
- A final object variable:
  - Only the reference (where it points in memory) is final
  - The values in the object can change via member methods
- We use constructors to create objects
Strings

- A String is a sequence of characters
- The `+` operator concatenates two Strings
- The `+=` operator appends a String
- First character has index 0
- A String can never be modified once created!
String methods

- length()
- substring()
- indexOf()
- lastIndexOf()
- charAt()
- trim()
- valueOf()
Logical expressions

- Logical expression has value either true or false

- Java has the boolean type with values **true** or **false**
Logical operators

- Three primary logical operators: **and (&&), or (||), not (!)**

- An **&&** operation is only true when both parts are true. A binary operation

- An **||** operation is true when either (or both) parts are true. A binary operation

- A **!** operation negates (switches) the value of the expression. A unary operation.
Equality

- Two equality operators for primitive types:
  
  
  `==` and `!=`

- When comparing objects, `==` compares the references, not the objects themselves

- Use the `.equals()` method, when available, to test for object equality

- Don't test floating point values for equality! Instead, test for “closeness” using relational operations:

  `<`, `>`, `<=`, `>=`
Ordering

- Ordering operators: <, >, <=, and >=. These only work on primitive types!

- Relational operators are the equality operators and the ordering operators

- For characters, ordering is based on the Unicode numbers of the characters
If decision statements

- An if statement has the form: if (expression) action

- An if-else statement has the form: if (expression) action1 else action2

- An if-else-if statement is used when there are many tasks to do, depending on the logical expressions
**Misc**

- **Short-circuit evaluation**: left side is evaluated first. If the result can be determined at that point, right side is not evaluated.

- `System.exit(0)` will terminate the program immediately.

- Use consistent indentation!
Switches

- A switch statement is useful instead of a long-winded if-else-if block

- Should always put either break at the end of a switch statement block, or a comment such as '//' FALL THRU'

- The default case means any case not matched by any of the other cases