Review

Spring 2016
CS 102
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
  - A low level language!

- **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 is 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

- **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:
  - 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 types (or other classes) as well as methods
  - A class is a template for an object

- Creating a variable/object from a type/class is called **instantiating** the type/class
Problem solving steps

- **Analysis**
  - What needs to be done?

- **Design**
  - How is it going to be done?

- **Implementation**
  - Make it so!

- **Testing**
  - Does it work correctly?
Readable programs

- **Comments** are English text
  - Single lines have a `//` before them in a Java file
  - `/* */`
  - `/** */`

- **Blank lines** make a program easier to read

- **Indentation** helps humans identify which code is within the method

- **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
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 methods have the following syntax:

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

Properties of the method

- Type that it returns
- A name for the method
- Any number (including zero) of parameters
- The body of the method (can be empty)

```
public static void main (String[] args) {
...
}
```
Java starts executing a program at the beginning of the main() method

Braces \{ \} are used to specify where a method 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

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
Escape sequences

- Java provides escape sequences for printing special characters
  - \b backspace
  - \n newline
  - \t tab
  - \r carriage return
  - \ \ backslash
  - \" double quote
  - \' single quote
Primitive variable types

- Java has 8 (or so) primitive types:
  - float
  - double
  - boolean
  - char
  - byte
  - short
  - int
  - long

  - real numbers
  - two values: true and false
  - a single character
  - integer numbers
Symbolic names vs. literal values

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

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

- It allows for changing the constant later on

- Are usually 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.

- Variables defined in a class but outside a method are initialized to a default value.

- Variables defined in a method are **not** initialized to a default value.
Math

- Standard operators: + - * / 
- Note that / can be either integer division or floating-point division 
- % computes the remainder (aka modulus) 
- Can provide numbers in decimal or scientific notation
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 operator has two operands
  - Example: 3+4, 6*3, etc.
  - Left one is evaluated first

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

- Operators have precedence
  - For example, * and / are evaluated before + and -
Overflow

Consider:
```java
byte b = 100;
b = b * 100;
```

A byte can only hold up to +127

When numbers in a byte get too big, it is called **overflow**
- Java does not tell you that this happened!

**Underflow**: b -= b*100;
Operators

- Assignment: =
- Increment (++) and decrement (--)

Consider:

```java
int i = 5;
System.out.println (i++);
System.out.println (++i);
System.out.println (i);
```

There are 4 ways to add 1 to an int:

- `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);

  double d = 3.4;
  System.out.println ((int) d);
  ```
Creating one:

```java
Scanner stdin = new Scanner(System.in)
```

Don’t use `Scanner.create()`!

Methods:

- `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 that 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 values either true or false
- Java has the boolean type with values true or false
- Truth table: method to dissect a logical expression
Logical operators

- Three primary logical operators: **and**, **or**, **not**
- An **and** operation is only true when both parts are true
- An **or** operation is true when either (or both) parts are true
- A **not** operation negates (switches) the value of the expression
- Logical operators: and is `&&`, or is `||`, not is `!`
- Not operator is unary
- If the first boolean expression in an `&&` statement is false, then the rest of the expression is skipped. This is called **short circuited**:

  ```java
  if ((x > 0) && (3 / x == 1)) //the second part is not executed if the first part returns false
  ```
Equality

- Two equality operators: `==` 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”
Ordering

- Ordering operators: <, >, <=, and >=. These only work on primitive types!
- Relational operators are the equality operators and the ordering operators
- For booleans, false is less than true
- For characters, ordering is based on the Unicode numbers of the characters
If 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
- A for-each expression is like an if-else but is used to iterate over all the items in an ordered set.

    (count == 5) ? count++ : count--;


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

```
// FALLING THRU
```

The `default` case means any case not matched by any of the other cases.
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!