ICS 111
Final Review 1/2

- Basic types
- Expressions, variables, and assignments
- Sequence
- Repetition
- Conditionals
- Parallel execution
- Program structure

ICS 111 review: Java basic types

- byte, short, int, long
  - 1-byte, 2-byte, 4-byte, 8-byte integers
  - variable++ increments a numeric variable, **after** recording its value as the value of the expression
  - ++variable increments it before recording the value
  - also variable-- or --variable
- float, double
  - 4-byte, 8-byte floating point numbers
- boolean
  - true or false
- char
  - we haven’t looked at chars much, but we know strings are essentially immutable arrays of characters

ICS 111 review: Java Object types

- Byte, Short, Integer, Long, Float, Double, Boolean, Character
- Object and Object references
  - the `null` Object reference
  - `new` to create a new object and return the reference, after calling the constructor
- each class defines a new Object type
- Strings are Objects that store sequences of characters
- arrays are Objects that store any fixed number of elements of type Object or any basic type
- ArrayLists are like arrays, but can grow and shrink

ICS 111 review: Java operators and expressions

- arithmetic operators: +, -, *, /, %
  - if all the operands are integral, division and modulo use integer division, always rounding towards zero (i.e. truncating)
- boolean operators: &&, ||, !
- conditional expressions: b ? yes : no
  - yes and no must evaluate to the same type
- a variable by itself can be an expression
- many methods from the String and Math classes can be used in expressions

ICS 111 review: Java variables

- a variable is a name for a memory location that can remember a value
- local variables have scope from the declaration to the end of the block
- instance variables can be accessed by any instance method
- static variables can always be accessed: they are global variables
- public, protected, private (and default) modifiers
- instance and static variables are automatically initialized by Java, to default values
  - default values for basic types are 0 or `false`
  - default values for objects are `null`
  - there is no default initialization for local variables!

ICS 111 review: assignment

- the assignment `x = value` stores `value` into the memory named `x`
- initialization combines the syntax for assignment with the syntax for declaration

```
int x = value;
```

  - initializations are always required in ICS 111, and are almost always a good idea
- assignment of object variables copies the reference, and not the object
- calling a method performs an assignment to each of the parameters of the method

ICS 111 review: code execution in sequence

- statements are executed one after another
  - basic statements include assignments, and built-in primitives such as break, return, throw
  - method calls are also statements
  - declarations, with or without initialization, are also in sequence with statements
- sequences of statements are grouped into blocks
  - blocks are surrounded by { curly braces }
  - blocks may include local variable declarations
  - the body of a loop or a conditional may be a block, or a single statement
- logical (boolean) expressions with && and || use short-circuit evaluation, only evaluating the right-hand side if the value of the left-hand side makes it necessary
- if part of the evaluation of an expression throws an exception, the rest of the expression is not evaluated
  - what exactly is evaluated can be seen when expressions have side effects, for example count++, some method calls, or throwing exceptions
- sequence is so “normal” that most programmers don’t think much about it

ICS 111 review: repetition and loops

- three basic loop statements are for, while, and do..while
- `while (x < n) { …`
- `for (int x = 0; x < n; x++) { ...`
  - enhanced for is a kind of for loop, and for loops are particularly useful with arrays and Collections

    ```
    for (int x: a) { ...
    ```
- to avoid infinite loops, the body of the loop must make changes that affect the loop condition
- break exits from the nearest enclosing loop (or switch) statement
- recursion also allows repeated execution

ICS 111 review: conditionals

```
if (condition1) { … }
else if (condition2) { … }
else { … }
```

- there could be zero or multiple else if statements
- the else part is optional
- conditional execution is an essential part of computation, giving the ability to execute different parts of the code in different situations
- the boolean conditions may evaluate expressions, call methods, and so on

ICS 111 review: parallel execution: threads

- multiple threads in a graphics program:
  - the main thread executes as usual
  - the event dispatch thread thread executes in response to graphics events
    - this is an example of event-oriented programming)
    - the main thread may also schedule events for the event dispatch thread
- threads can also be started explicitly
  - we haven't studied this
  - useful for concurrent execution
    - e.g one thread might be waiting for input from a user
    - another thread might be waiting for input from a network socket
    - and another thread might be carrying out a background computation
  - threads can give higher performance if you have multiple CPUs (multiple cores)
- each thread has its own stack, so local variables are thread-specific, but threads share the global memory (the **heap**), which means that instance and global variables can be accessed by any thread
  - threads have to be careful when accessing shared variables
  - controlled concurrent access to shared variables is **synchronization**

ICS 111 review: parallel operations

- do the same operation to every element of an array
  - e.g. add 1 to every element of an array
  - or add 1 to every odd element of an array
  - or add the corresponding elements of two different arrays
  - this is a **map**, mapping an operation across the elements of an array
- summing all the elements of an array can also be done partly in parallel
  - in parallel, can add elements 1 and 2, elements 3 and 4, elements 5 and 6, elements 7 and 8, and so on
  - in the next step, add elements 1, 2, 3, 4, in parallel with adding elements 5, 6, 7, 8, and so on
  - this is a **reduce**, using the same operation to combine all the elements of the array
- simultaneous operations on many elements are often called **vector operations**
- we have not yet studied threads, and we have not yet studied MapReduce and vector operations

ICS 111 review: code structure

- naming conventions, comments and Javadoc
- code reuse through packages and inheritance
  - the less code we have to write, the fewer the bugs
- methods and classes help us to split code into understandable and manageable units
- controlling access to class variables helps when reasoning about code correctness
- when writing larger programs, all these come into play to make the program as clear and as bug-free as possible