Types, Memory and Exceptions

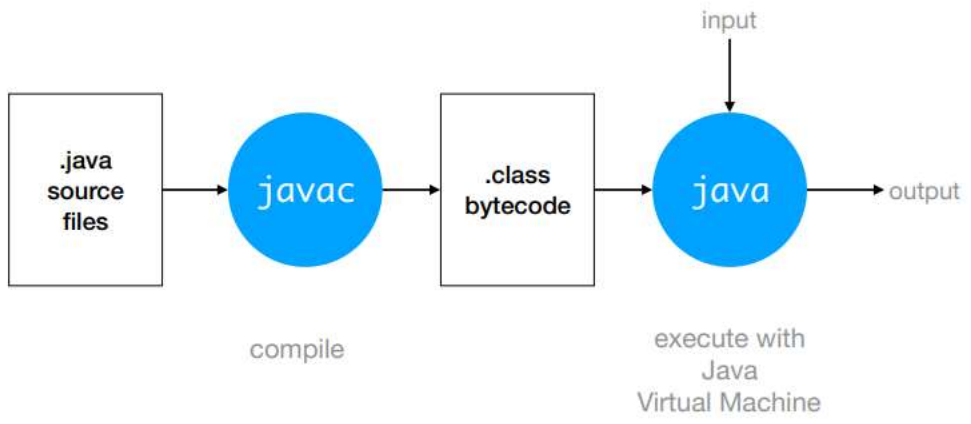
Friday, 26 January 2018 8:18 AM

Compiling and Running Files

- Compile = javac, run (interpret) = java

javac

- Compiles code into bytecode (executed by software interpreter)



- Other java compilers: ecj, gcj

gcc

- Produces native machine code (corresponding to hardware instructions)

Types

Type Conversion

- Widening reference conversion = allowed during compile and run time
    - (implicit) Assignment, method invocation
    - (explicit) toString()

Type Casting

- Narrowing reference conversion
    - Explicit type casting for compile time, may give error for runtime
        - `ClassCastException`
    - Has to be subtype -> supertype
    - Circle <: Shape
        - Subtype - implements/ extends

Variance of Types

- Is equals(Circle) <: equals(Object)? (no)
- Java arrays = covariant
    - S <: T, then S[] <: T[]

Memory

JVM partitions memory

- Method area: code for methods

- Metaspace: meta information about classes
- Heap: stores dynamically allocated objects
    - JVM garbage collector checks for unreferenced objects
- Stack: stores variables (primitive + reference)

`null`
- Reference variables point to null at first (initialisation)

Call Stack

JVM creates a stack frame for each method call (destroyed after return)
- `this`
    - Only for instance method calls
- Method arguments
- Local variables
- Primitive types = call by value, Reference types = call by reference
    - Have to use Wrapper class to keep primitive values modified
        - Else, only the values will be copied

Exceptions

- Error = cannot recover (OutOfMemoryError, heap), (StackOverflowError, stack)

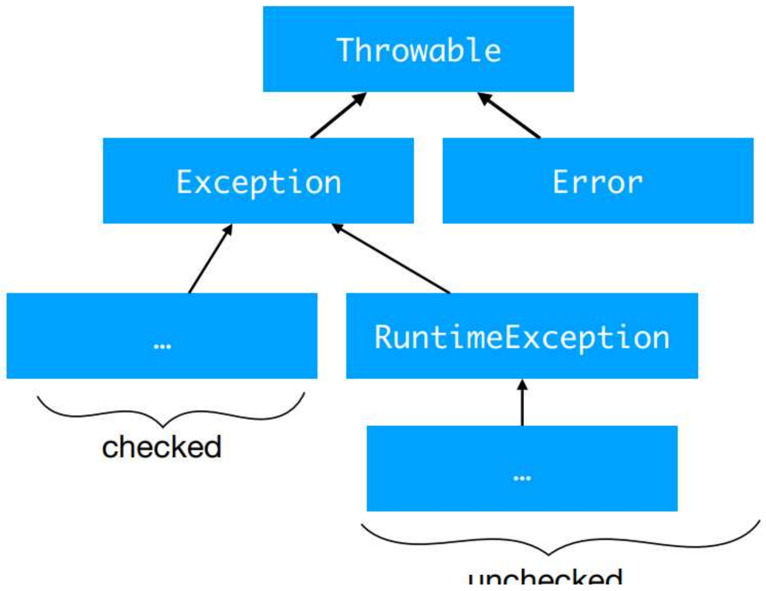
`try-catch-finally`
- Uncaught exceptions passed to calling methods (exception propagation)

`catch (ExceptionA | Exception B)`
- Multiple exceptions to be handled in the same way
- `System.err.println("");`

`finally`
- Executed even with return/ throw in catch block

`throw`
- Throw exception to calling function
- Cannot be used by main()

Checked vs. Unchecked Exceptions



- Checked Exceptions: should be caught
    - FileNotFoundException
- Unchecked Exceptions: subclasses of RuntimeException, or Error
    - InputMismatchException, NoSuchElementException, IllegalArgumentException

Overriding

- Have to throw the same/ more specific/ no checked exception
- Cannot declare checked exception not in super class

Good Practices

- Catch and re-throw exceptions (deallocation of memory resources in finally)
- `catch (Exception e) {}`
    - Too generic, exceptions silently ignored
- `System.exit(0)`
    - Overreacting
- Handle implementation-specific exceptions within abstraction barrier

Exercises

1. Type Conversion
    - Line 1: widening reference conversion, allowed
    - Line 2: ArrayStoreException (runtime exception)
4. Primitive type conversion
    - Part A, Line 3: incompatible types: possible lossy conversion from double to int
    - Part B, incompatible types: boolean cannot be converted to int and int cannot be converted to boolean
    - Part C, Line 9: ClassCastException (narrowing reference conversion)
    - Part D
        - Line 7: I is abstract; cannot be instantiated
        - Line 9: incompatible types, I cannot be converted to A
    - Part E
        - Line 14: incompatible types, J cannot be converted to I (unless explicit cast as in Line 15)
        - Error in Lines 16, 17 (unless explicit case as in Lines 18, 19
    - Part F, explicit casting necessary (for implements/ extends relationships)
    - Part G
        - Line 12: incompatible types
        - Line 14: explicit casting only allowed if type of variable is possible interface/ parent class
5. Exception Handling
    - Part A, Line 6: error, unreachable statement
    - Part B, Line 5: after catching error in f, error not caught in main
    - Part C, Line 4: error, unreported exception; must be caught or declared to be thrown
    - Part D
        - Line 4: only have to declare throwing a more general/ same exception
        - Line 5: error is caught and handled in f
    - Part E, Line 4: error caught in main
    - Part G, Line 7: error, exception has already been caught (in Line 5)
    - Part H, Line 5: AIOOB caught
    - Part I, Line 15: error caught in main

- Remaining statements not executed (Line 14)