Abstraction and Encapsulation

Monday, 15 January 2018 10:09 AM

Program = instructions + data (in memory)
Compiler & interpreter translates to assembly/ machine code

Abstraction

Abstraction Principle: always abstract common parts

Type & Variables

Dynamic vs. Static types

- Java = static type, have to declare variables
- Java is a type-safe language (compiler needs to know how much memory to allocate for each variable)

Reference type vs. Primitive types

Variable name stores value in memory
Pointer to variable stores address of location

Composite Data Types

E.g. Circle: many different representations

- Centre + Radius/ Diameter
- Corners

Functions

Separation of concerns, reusability
Implementation vs. Use: maintaining the abstraction barrier

Encapsulation: Classes & Objects

Methods & fields
Above abstraction barrier: call provided interface to use composite data type

Constructor: initialises object
Accessor (getter): retrieve properties
Mutator (setter): modify properties

`static`

- Associate method/ field with class and not an actual object (instance)
- Class vs. Instance (object) fields and methods
- Only has one copy

`jshell`

- Provides a read-evaluate-print loop (REPL)

Exercises

1. Only one copy of the class fields exists
3. Line 14: B cannot access the private fields of A