

Core Java

7. Classes: References, Access, Nesting

References

- Variables of Classes are references to objects in memory
- Pointers without the *
- Objects are passed by reference to functions
- Static references refer to single instance in memory
- Discuss Memory Layout

Mutability

- Need to be careful when passing objects around
- Objects are modifiable unless they prevent modifications
- String is immutable
- Is immutability a good practice?

null

- *null* refers to a non-reference location in memory
- Object references are *null* if not created
- This can be used to set/deset uninitialized values
- What about objects which had a reference and then lost them?

this

- *this* within a class, refers to the current object
- Sometimes, require to pass the current instance 'back'
- Examples are two-way lists

Access

- Specifies how the class can be accessed by other classes
- public, protected, friendly*
- private only for variables/functions/nested classes
- public, accessed by all
- protected, accessed by all subclasses and in same package
- friendly, accessed in same package only

Nesting

- A class can be nested within a class
- Not usually used, but may give certain benefits
- Allows more access specifiers
- Can be private, scope of the class
- Can also be static: create instances without creating an instance of outer class
- Note: Anonymous classes are similar

Demonstration

- Compile and Execute a few programs

Questions?