Theory and Specification

Principles:

1. It must be simple, object-oriented, and familiar.

2. It must be robust and secure.

3. It must be architecture-neutral and portable.

4. It must execute with high performance.

5. It must be interpreted, threaded, and dynamic.

Access Modifiers:

- public

    - any class can access
    - accessible by entire application

- private

    - only accessible within the class

- protected

    - allow the class itself to access them
    - classes inside of the same package to access them
    - subclasses of that class to access them

- package protected

    - default
    - the same class and any class in the same package has access
    - protected minus the subclass unless subclass is in package

- Static: Belongs to class not an instance of the class

Type Classifications:

- Concrete Types

- concrete types describe object implementations, including memory layout and the code executed upon method invocation
- the exact class of which an object is an instance not the more general set of the class and its subclasses
- beware of falling into the trap of thinking that all concrete types are single classes!
- Set of Exact Classes
- ex: List x has concrete type ArrayList, LinkedList, ...

- Abstract Types

  - Abstract types, on the other hand, describe properties of objects
  - They do not distinguish between different implementations of the same behavior
  - Java provides abstract types in the form of interfaces, which list the fields and operations that implementations must support

Generics:

- Definition

  - generics are a facility of generic programming
    * a style of computer programming in which algorithms are written in terms of types to-be-specified-later that are then instantiated when needed for specific types provided as parameters
  - ex: compiletime:

    ```
    List<String>
    ```

    runtime: List

- Notes

  - in java, generics are only checked at compile time for type correctness
  - generic type information is then removed via a process called type erasure, to maintain compatibility with old JVM implementations, making it unavailable at runtime

- Sources

  - https://en.wikipedia.org/wiki/Generics_in_Java

Sources:

- https://en.wikipedia.org/wiki/Java_(programming_language)