

# 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.

---

## Basic Definitions:

---

- an object is a runtime entity and its state is stored in fields and behavior is shown via methods
  - methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication
  - the Object class is the parent class of all the classes in java by default
- a class represents the set of properties or methods that are common to all objects of one type
  - a class can contain fields and methods to describe the behavior of an object

---

## 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](https://en.wikipedia.org/wiki/Generics_in_Java)

---

### Sources:

---

- [https://en.wikipedia.org/wiki/Java\\_\(programming\\_language\)](https://en.wikipedia.org/wiki/Java_(programming_language))
- [https://www.tutorialspoint.com/java/java\\_interview\\_questions.htm](https://www.tutorialspoint.com/java/java_interview_questions.htm)