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 it's 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
- an interface is an abstract type that is used to specify a behavior that classes must implement

Inheritance:

- the class which inherits the properties of other is known as subclass (derived class, child class)
- the class whose properties are inherited is known as superclass (base class, parent class).
- extends is the keyword used to inherit the properties of a class

```
class Super {
   .....
   .....
}
class Sub extends Super {
   .....
   .....
}
```

Overloading:

occurs when two or more methods in one class have the same method name but different parameters

```
class Dog{
    public void bark(){
        System.out.println("woof ");
    }
    // overloading method
    public void bark(int num){
        for(int i=0; i<num; i++)
            System.out.println("woof ");
    }
}
```

Overriding:

- overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes
- final methods can not be overridden
- you can call parent class method in overriding method using super keyword

Polymorphism:

- polymorphism is the ability of an object to take on many forms
- the most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object

Abstraction:

- abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user
- abstraction is achieved using abstract classes and interfaces

Encapsulation:

- encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit
- the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class, also known as data hiding
- encapsulation in Java
  - declare the variables of a class as private
  - provide public setter and getter methods to modify and view the variables values

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)`
- `https://www.tutorialspoint.com/java/java_interview_questions.htm`