Packages & Wrapper Classes

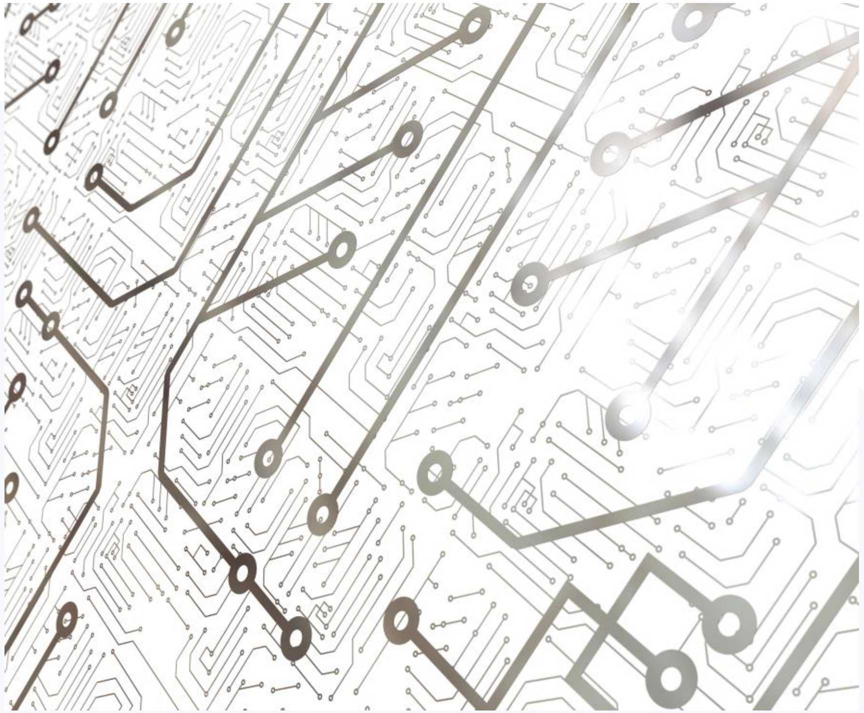
Packages

Packages are used to organize classes and interfaces into namespaces. They help in managing and structuring code by grouping related classes together. Classes and interfaces within a package share a common namespace, reducing naming conflicts and enhancing code organization and maintainability

Packages can also serve as access protection mechanisms. Classes and members marked with the default (package-private) access modifier are accessible only within the same package, providing a level of encapsulation and access control.

Import & Static Import

- To define a package in Java, you include a package statement at the beginning of your source file, specifying the package name. For example, package com.example.myapp; declares that the classes in the file belong to the com.example.myapp package.



Defining a Package

- "Import" and “Static Import" are used to access classes, methods, and constants defined in other packages without specifying their fully qualified names.
- "Import" is used for classes and interfaces, while "static import" is used for static members (e.g., constants and methods) of classes.
- "Import" reduces the need for fully qualified class names, making code more readable.

Wrapper Classes

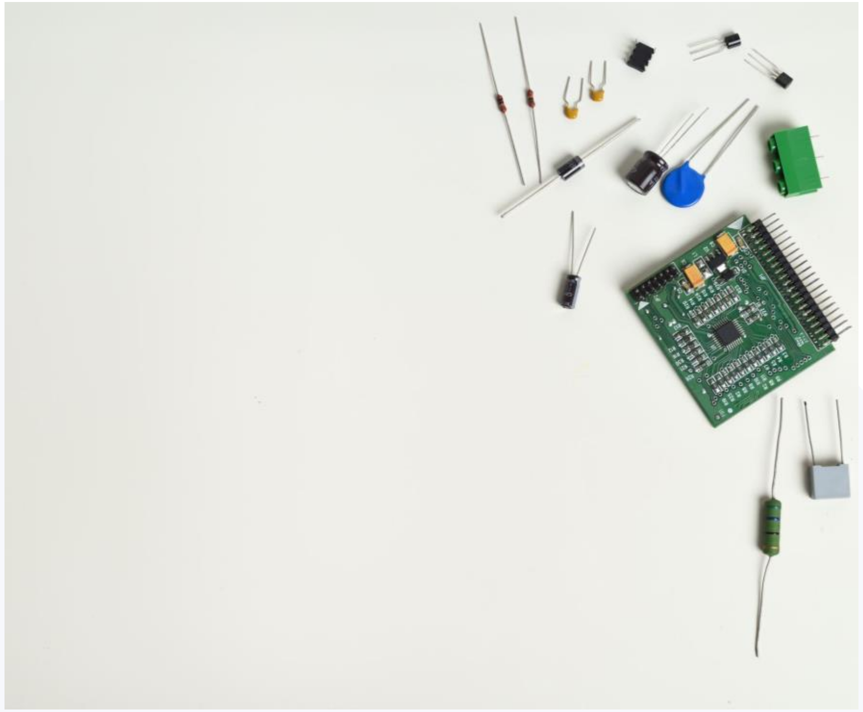
- Wrapper classes are classes that encapsulate primitive data types (e.g., int, char) and provide additional methods and functionality. They allow primitive types to be treated as objects and are used in scenarios where objects are required, such as collections and generics.

Why Wrapper Class

- Wrapper classes are used to convert primitive data types into objects, allowing primitive types to be treated as objects. This is useful in scenarios where objects are required, such as when working with collections, generics, and other Java APIs that expect objects rather than primitive types. Wrapper classes provide a bridge between the world of primitives and the world of objects, enabling greater flexibility and functionality in Java programming.

Handling Wrapper Class

- To handle wrapper classes, you can create instances of the wrapper classes to wrap primitive values, access their methods and attributes, and perform conversions between primitive types and wrapper objects. Additionally, you can use autoboxing and unboxing, which automatically convert between primitives and their corresponding wrapper objects.



xebia.com