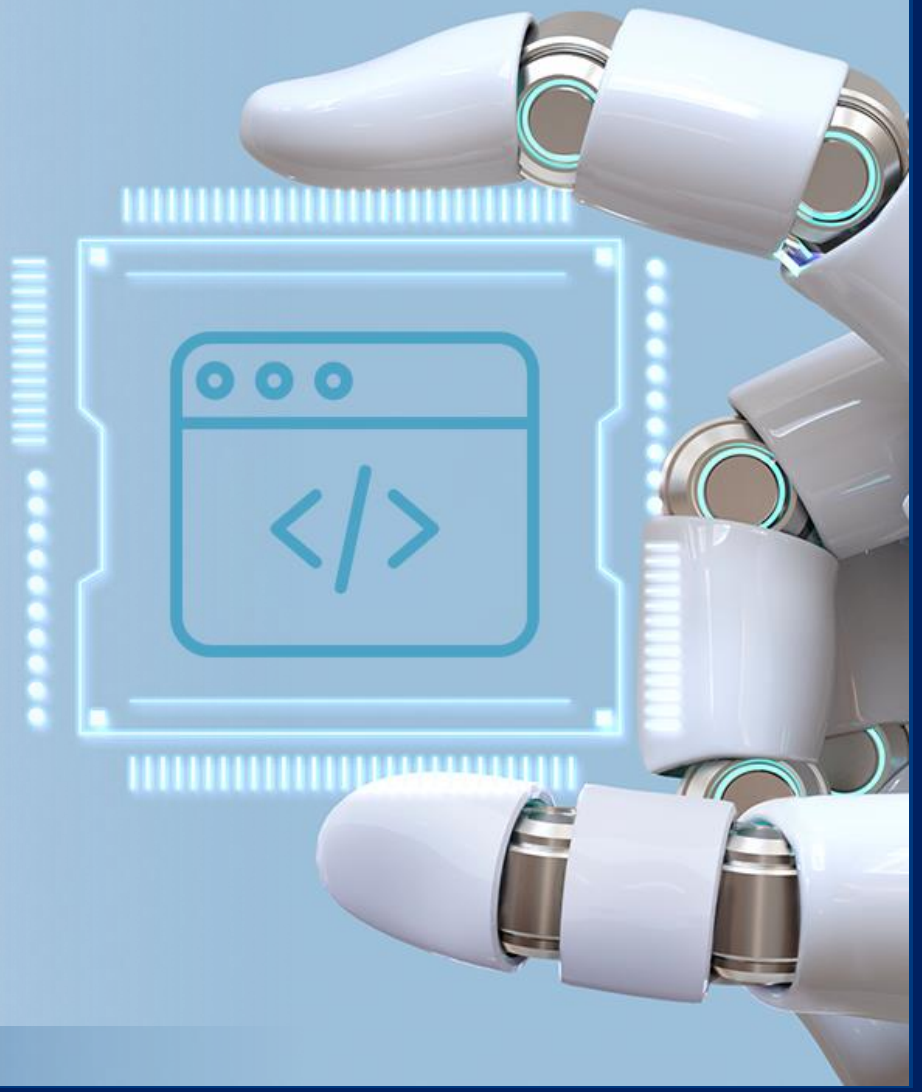
edunet
foundation

Java OOP Concepts



Disclaimer

The content is curated from online/offline resources and used for educational purpose only

edunet foundation

Agenda

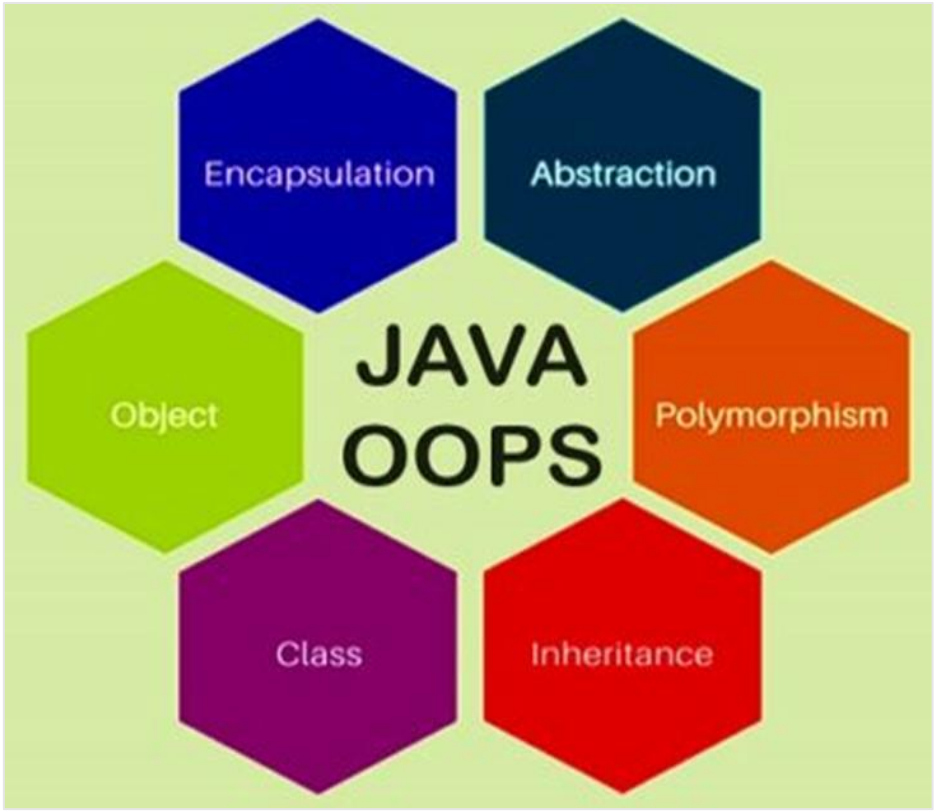
You will learn in this lesson:

- Revisiting Java OOP Concepts
- Java Object and Class
- Java Packages
- Methods
- Constructor
- Access Modifiers, and Non-Access Modifiers
- Abstraction
- Encapsulation
- Inheritance
- Polymorphism
- Interfaces

Introduction

Welcome everyone to the presentation on Java OOP Concepts

- Object-Oriented Programming (OOP) is programming around the concept of objects, which can be thought of as real-world entities with their own characteristics (attributes) and behaviors (methods).
- OOP provides a structured and modular approach to software development, allowing programmers to organize code into reusable and understandable components.
- The fundamental principles of OOP guide the design and implementation of object-oriented systems, enabling developers to create robust, flexible, and maintainable code.



Source: https://www.techyv.com/article/what-do-you-mean-oops-concept-java/

Revisiting Java Object-Oriented Paradigms

- After 27 years of existence, Java is still doing well.
- Programmers who know it are still in high demand.
- They will continue to be sought after for a long time to come as over 90% of the Fortune 500 companies still rely on Java for their development projects.

Source: https://dev.to/tech_sam/let-s-revisit-java-in-2019-3o8c

Understanding OOP concepts through core Java

OOP concepts include:

- Abstraction
- Encapsulation
- Inheritance and
- Polymorphism

Source: https://www.techyv.com/article/what-do-you-mean-oops-concept-java/

Java Class and Objects

- Java Class
  - A class is a blueprint for the object. Before we create an object, we first need to define the class.
- Java Objects
  - An object is called an instance of a class.

```
class MyClass {
  // class attributes
  // constructor
  // methods
}

public class Main {
  public static void main(String[] args) {
    //Create an object of MyClass
    MyClass myObject = new MyClass();
}
}
```

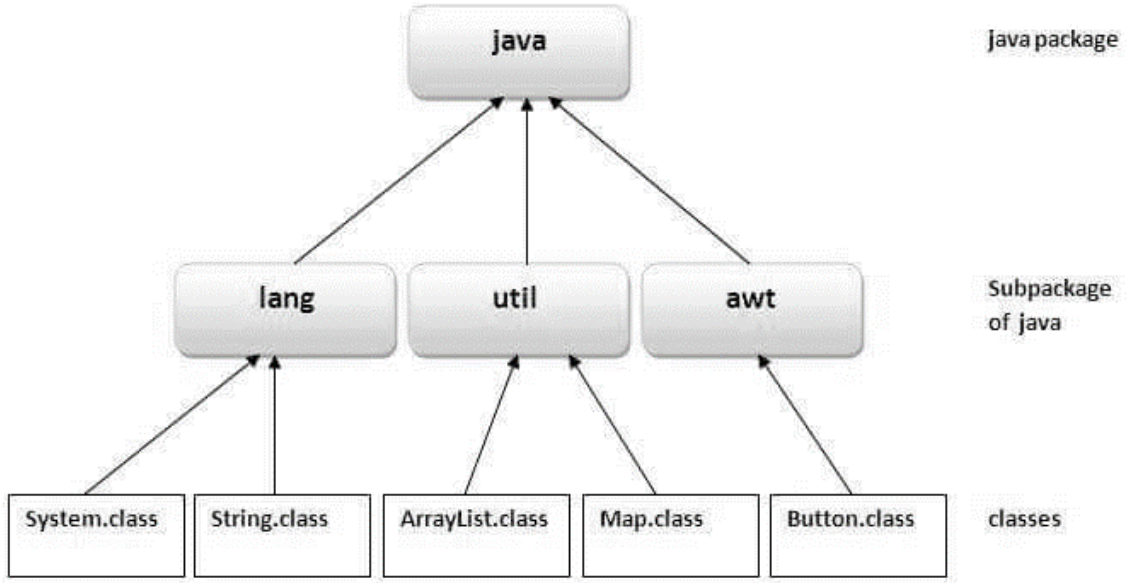
Source:

Java Packages

- A Java package is a group of similar types of classes, interfaces, and sub-packages.
- Packages in Java provide a way to organize and manage classes and interfaces.
- Package in Java can be categorized in two forms, **built-in package**, and **user-defined package**.

Advantages:

- Used to categorize the classes and interfaces.
- Provides access protection.
- Removes naming collision.



Source: https://www.javatpoint.com/package

edunet foundation

Lab Exercise

Hands On - 1: Basic example to create class and object

Java Interfaces

- An interface is a programming construct that defines a contract or a set of methods that a class must implement.
- It acts as a blueprint for classes, specifying the methods they should have without providing the implementation details.

Source: https://www.javatpoint.com/interface-in-java

Java Methods

- A method is a block of code that performs a specific task.
- Dividing a complex problem into smaller chunks makes your program easy to understand and reusable.

In Java, there are two types of methods:

Syntax:

```
public void methodName(parameter1, parameter2, ...) {
// body of the method
}
```

Lab Exercise

Hands On - 2: Demonstrate the use defined method to calculate a total of 2 numbers in Java

Hands On - 3: Demonstrate the methods with parameters in jav

Java Constructors

A constructor in Java is like a method that is invoked when an object of the class is created.

Unlike Java methods, a constructor has the same name as that of the class and does not have any return type.

edunet foundation

Java Constructors

Types of Constructor

In Java, constructors can be divided into 3 types:

- No-Arg Constructor
- Parameterized Constructor
- Default Constructor

```
public class MyClass {

  // class attributes
  // methods

    // constructor
  public MyClass(String name) {
    this.name = name;
  }


}
```

edunet foundation

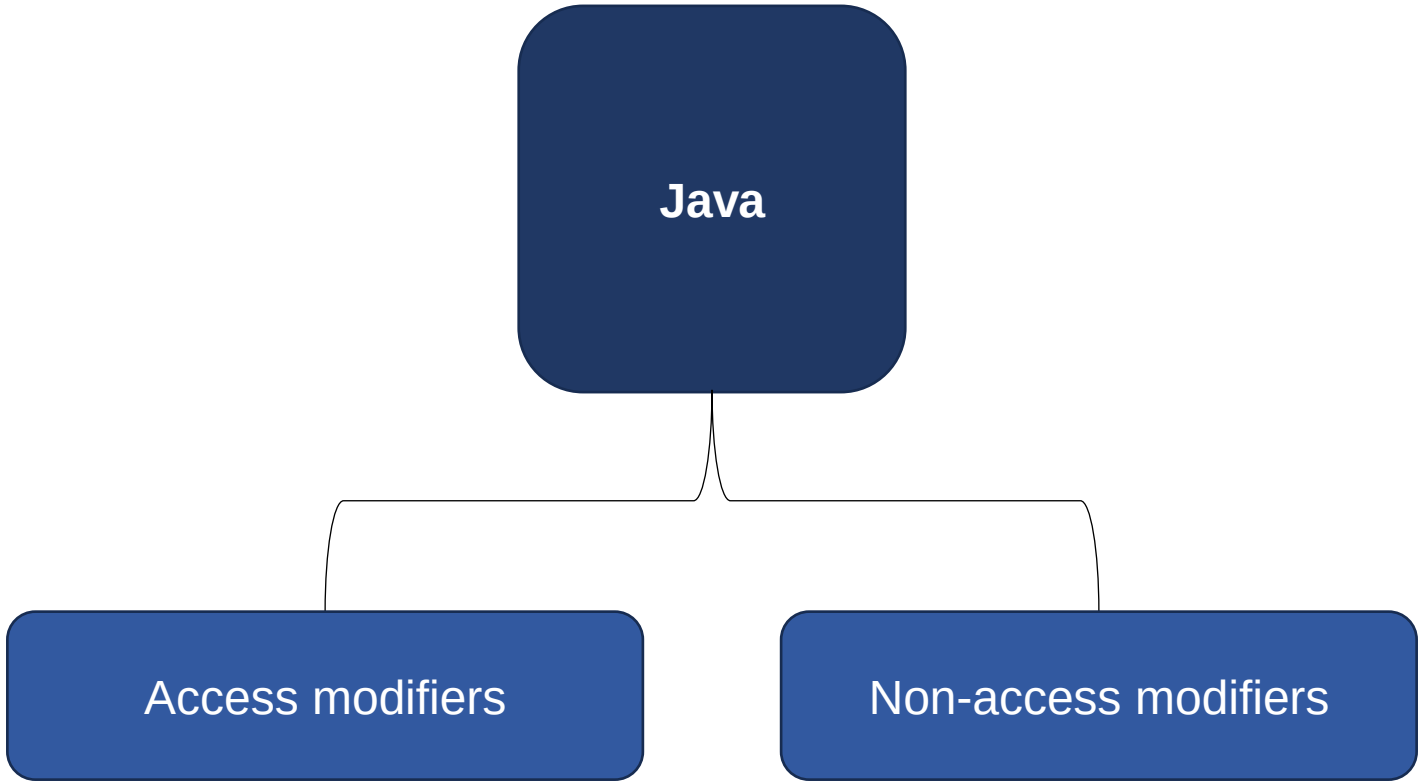
Lab Exercise

Hands On - 4: Demonstrate constructor and constructor with parameters in Java

Access Modifier

- The access modifiers in Java specify the accessibility or scope of a field, method, constructor, or class. We can change the access level of fields, constructors, methods, and classes by applying the access modifier to them.

There are two types of modifiers in Java:

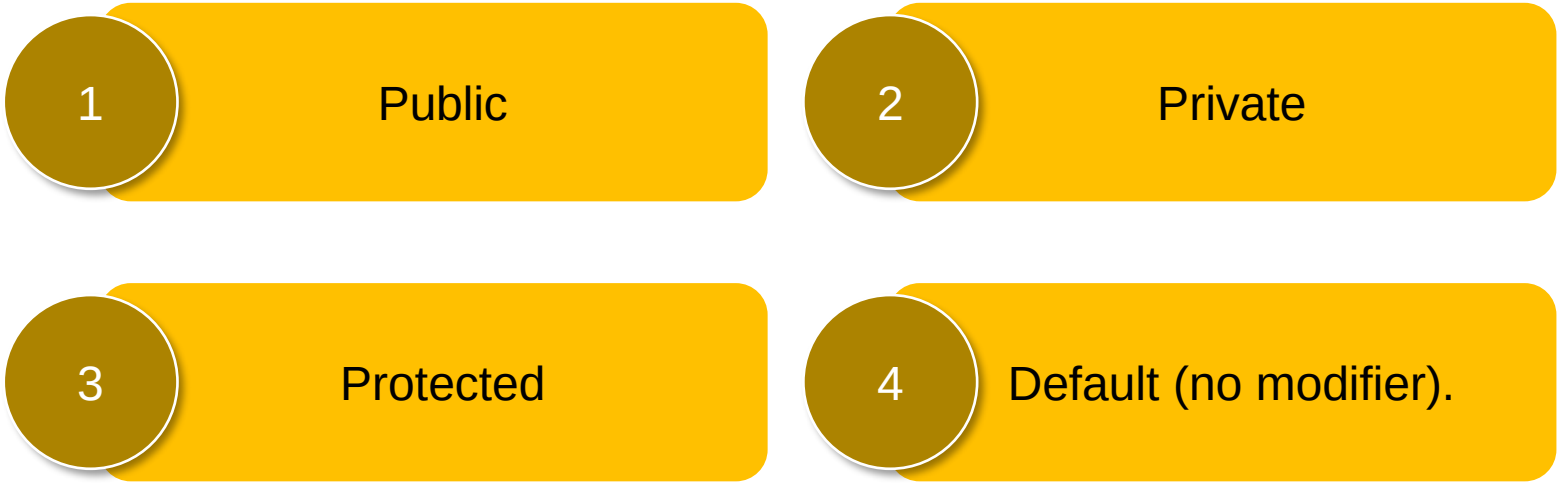


Access Modifier

Access Modifier

- An access modifier in Java determines the level of visibility or accessibility of a class, method, variable, or constructor.
- It controls which parts of the code can access or interact with these elements.

There are four access modifiers:



Access Modifier

Non-access modifiers

- A non-access modifier in Java provides additional characteristics or behavior to classes, methods, variables, or constructors.
- These modifiers do not control accessibility but modify the behavior or properties of the elements.

Examples of non-access modifiers include **final**, **abstract**, and **static**.

- They add specific features or constraints to the elements without affecting their accessibility.

Return Type

- The return type refers to the data type of the value that a method or function returns.
- It specifies what type of data the method will produce as its output or result.
- The return type is declared in the method signature and is defined using the appropriate data type keyword.

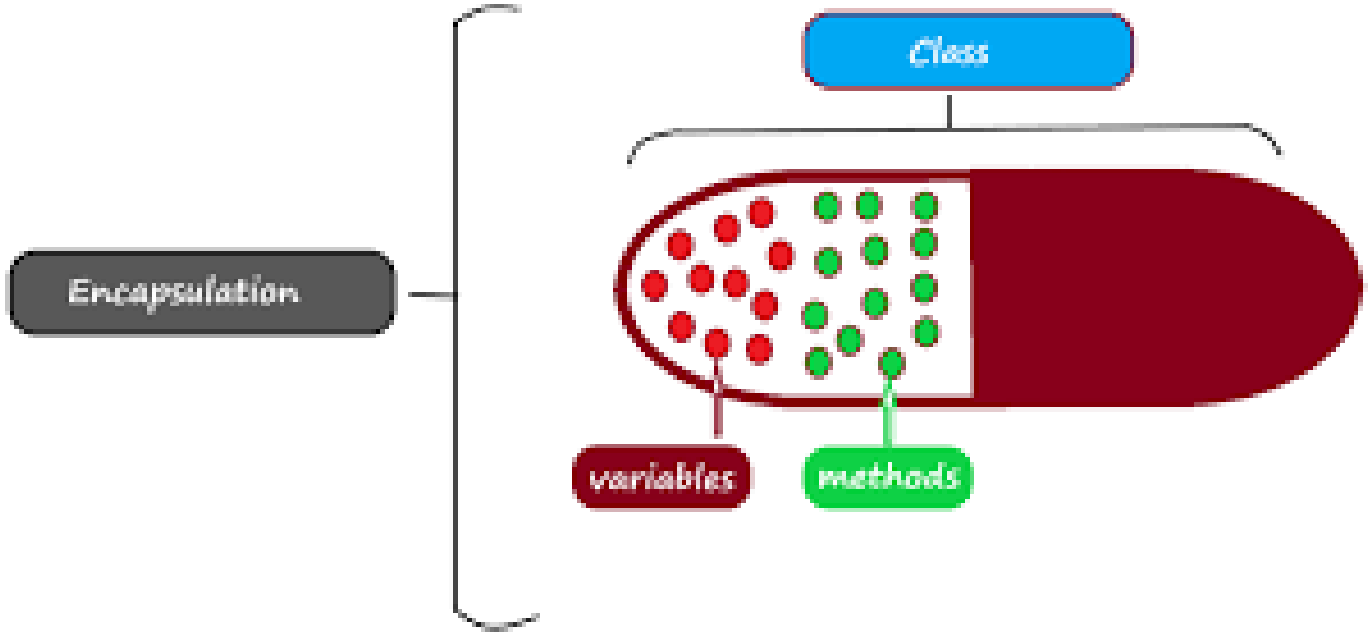
Lab Exercise

Hands On - 5: Demonstrate the various access modifiers in java

Hands On - 6: Demonstrate the various non access modifiers in java

Encapsulation

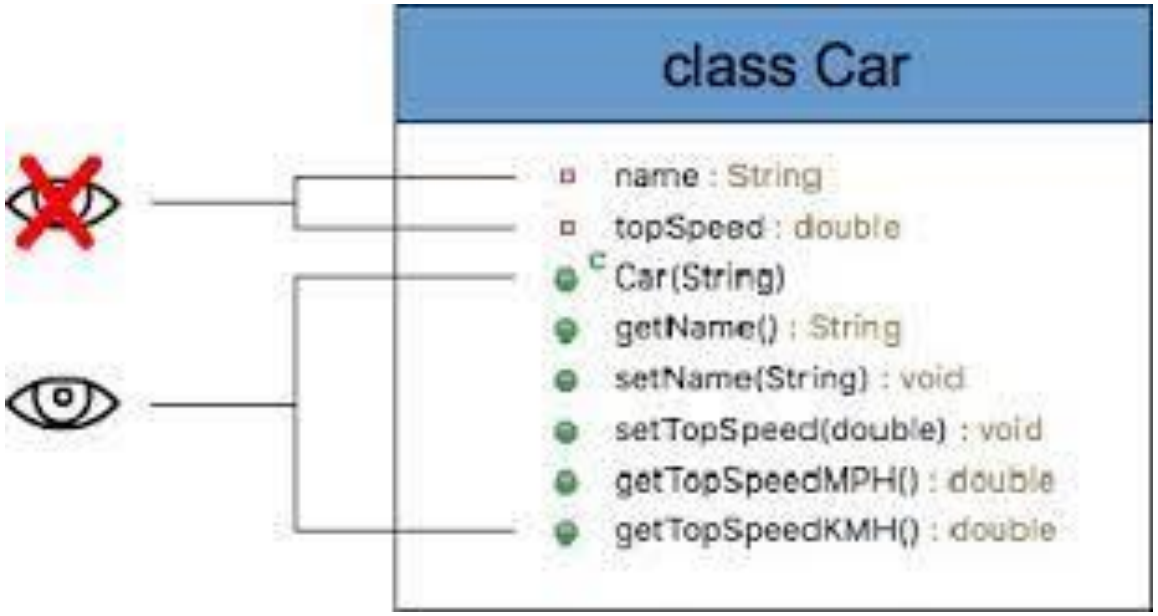
- Encapsulation is a protective barrier that keeps the data and code safe within the class itself.
- We can then reuse objects like code components or variables without allowing open access to the data system-wide.



Source: https://techblogstation.com/java/oops-concepts-in-java/

How Encapsulation Works?

- Encapsulation lets us reuse functionality without compromising security.
- It’s a powerful, time-saving OOP concept in Java.
- For example, we may create a piece of code that calls specific data from a database.



Source: https://javatutorial.net/java-encapsulation-example/

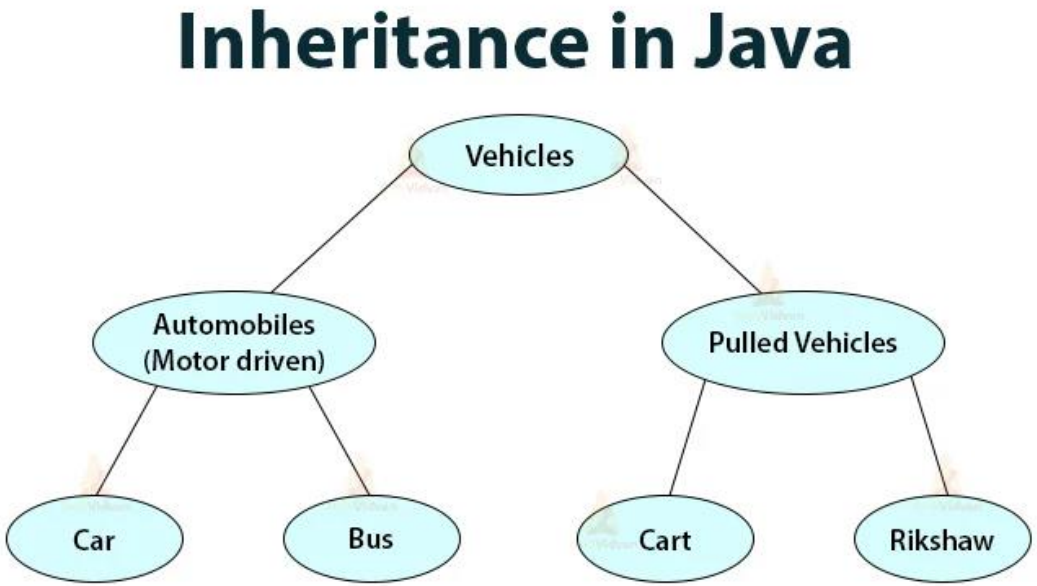
edunet foundation

Lab Exercise

Hands On - 7: Demonstrate encapsulation in java

Inheritance

- Inheritance lets programmers create new classes that share some of the attributes of existing classes.
- Using Inheritance lets us build on previous work without reinventing the wheel.

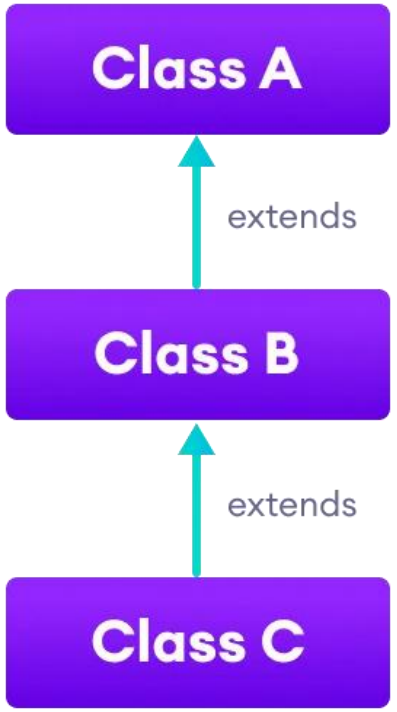


Source: https://techvidvan.com/tutorials/java-inheritance/

How Inheritance Works?

Let's a new class adopt the properties of another.

- Inheriting class is called a subclass or a child class.
- The original class is often called the parent.
- Keyword extends is used to define a new class that inherits properties from an old class.

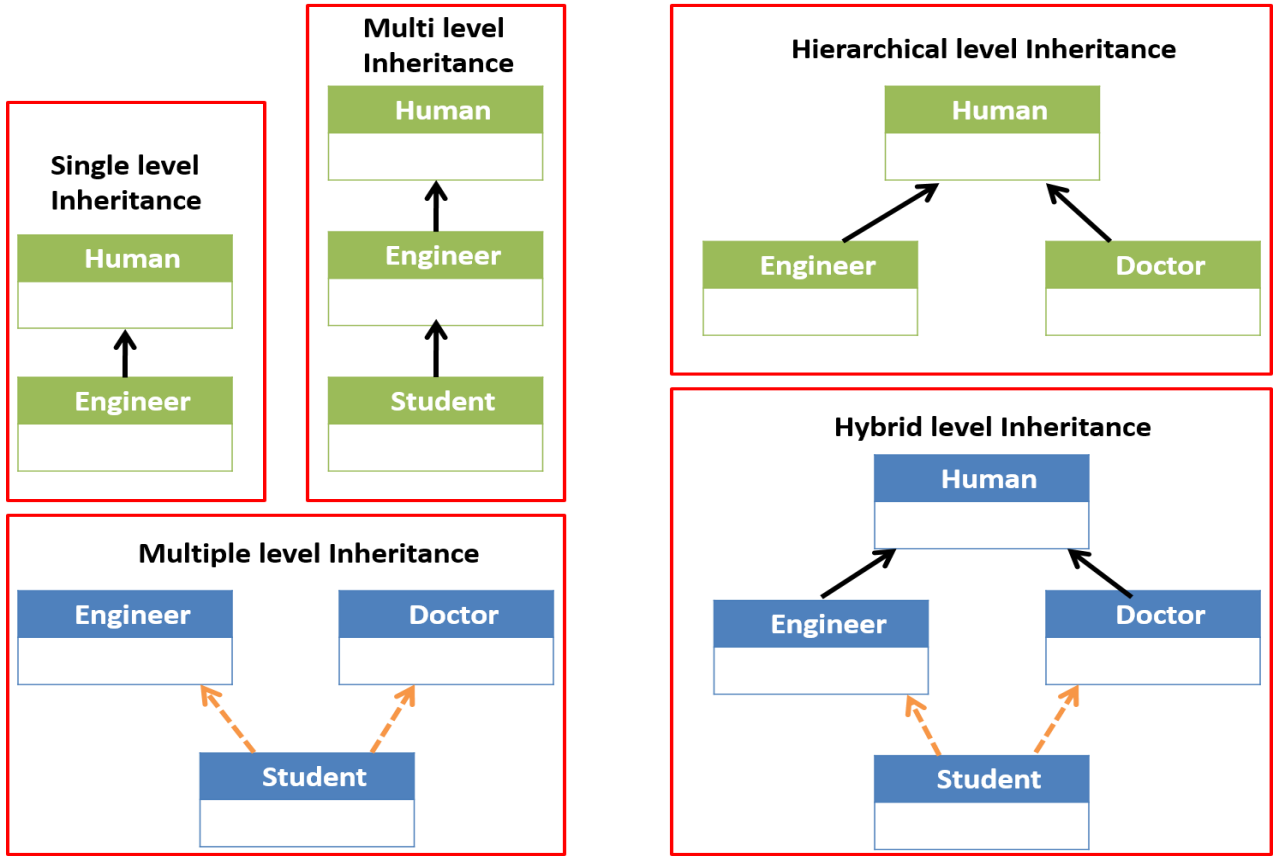


Source: https://www.programiz.com/java-programming/inheritance

Types of Inheritance

There are five types of inheritance.

- Single
- Multiple
- Multilevel
- Hierarchical
- Hybrid



Source: https://bytesofgigabytes.com/java/inheritance-in-java/

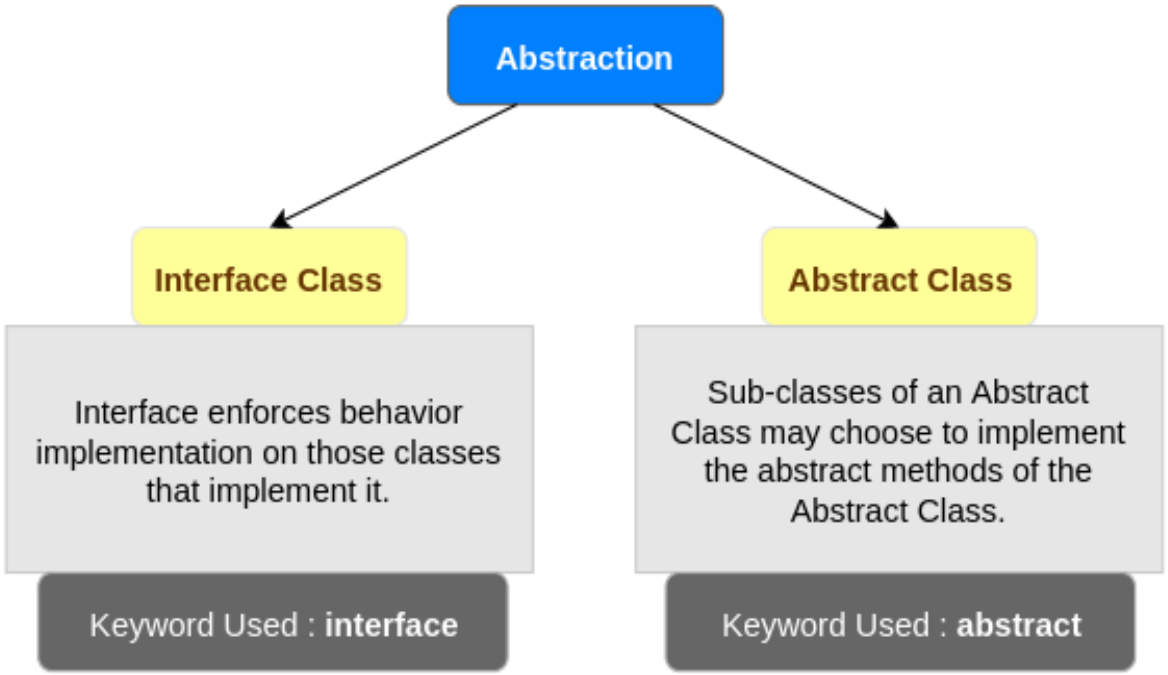
edunet foundation

Lab Exercise

Hands On - 8: Demonstrate inheritance in java

Abstraction

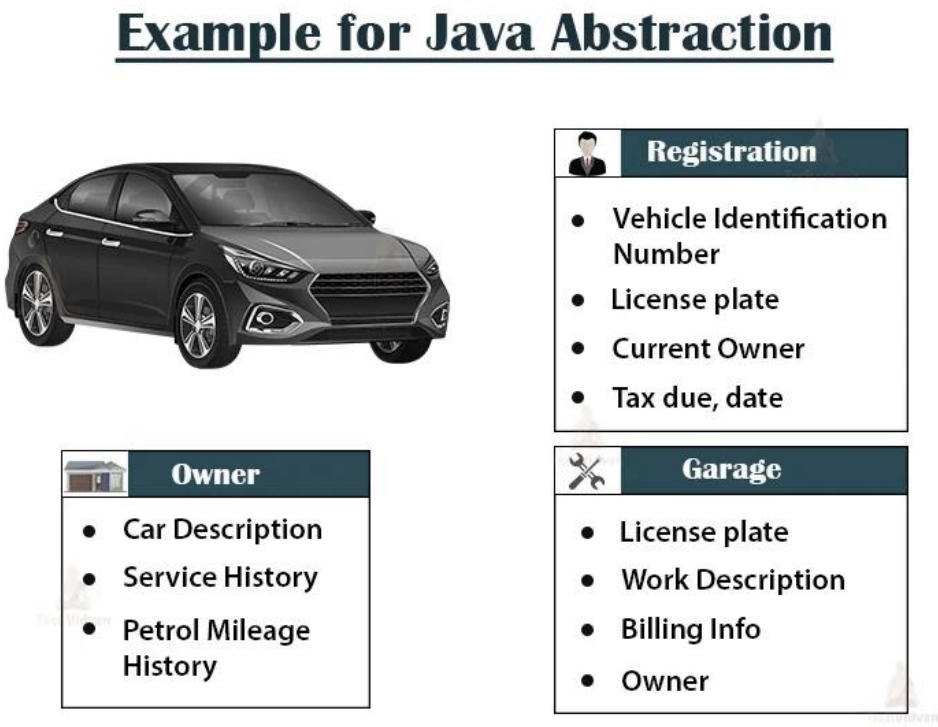
- In Java, abstraction means simple things like objects, classes and variables represent more complex underlying code and data.
- It lets you avoid repeating the same work multiple times.



Source: https://www.tutorialkart.com/java/abstraction-in-java/#gsc.tab=0

How Abstraction Works?

- Abstraction lets programmers create useful and reusable tools.
- For example, a programmer can create several different types of objects, which can be variables, functions or data structures. Programmers can also create different classes of objects as ways to define the objects.



Source: https://techvidvan.com/tutorials/abstraction-in-java/

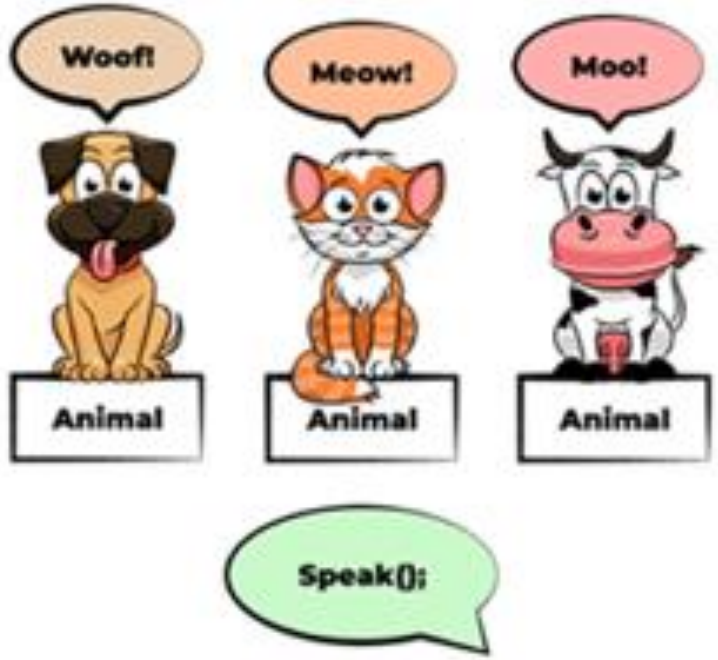
edunet foundation

Lab Exercise

Hands On - 9: Demonstrate abstraction in java

Polymorphism

- Allows programmers to use the same word in Java to mean different things in different contexts.
- One form of polymorphism is method overloading.
- The other form is method overriding.



Source: https://codegym.cc/groups/posts/polymorphism-in-java

How Polymorphism Works?

- We might create a class called “**horse**” by extending the “**animal**” class.
- That class might also implement the **“professional racing”** class.
- The “**horse**” class is “**polymorphic**,” since it inherits attributes of both the “**animal**” and **“professional racing”** class.

-> A Boy behave like a Student in a School

-> A Boy behave like a Customer in Market or Shopping Mall

-> A Boy behave like a Passenger in a Bus

-> A Boy behave like a Son in Home

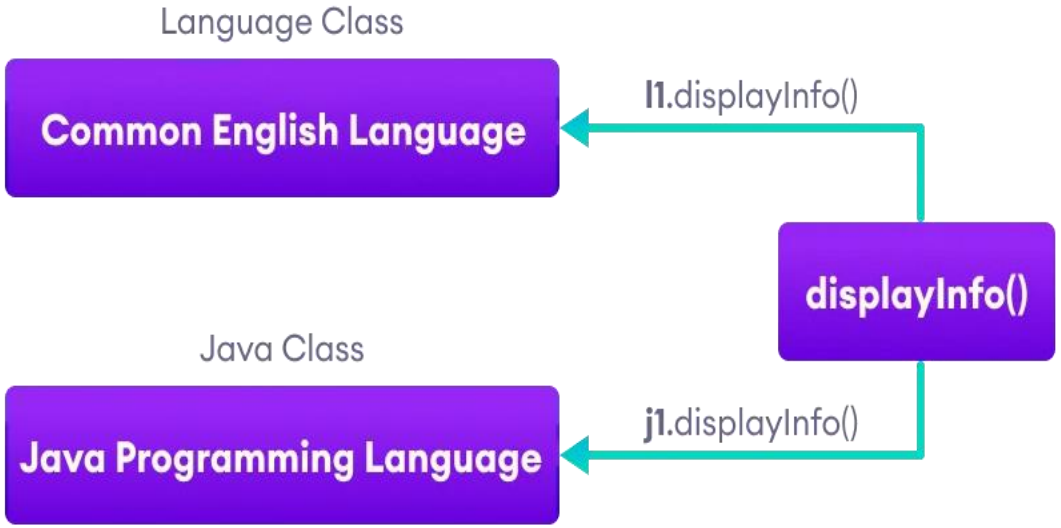
Source: https://www.javaguides.net/2018/08/polymorphism-in-java-with-example.html

Java Polymorphism

- Polymorphism is an important concept of object-oriented programming.
- It simply means more than one form.
- Polymorphism allows us to create consistent code.

We can achieve polymorphism in Java using the following ways:

- Method Overriding
- Method Overloading

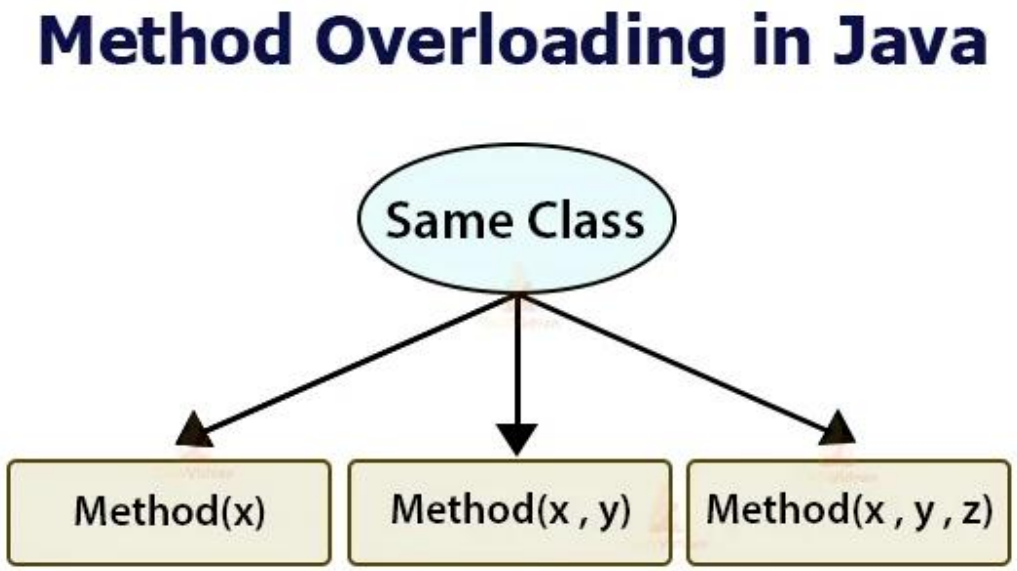


Working of Java polymorphism

Source: https://www.programiz.com/java-programming/polymorphism

Method Overloading

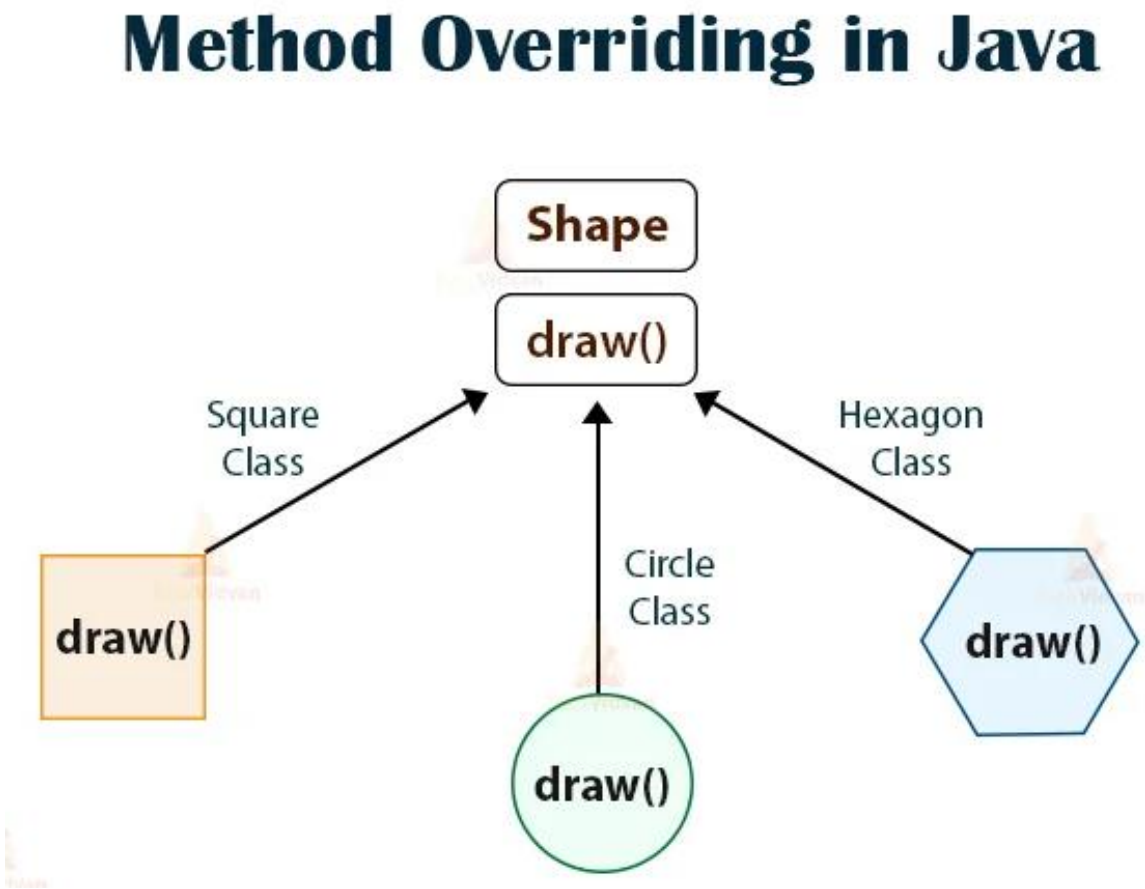
- A single method may perform different functions depending on the context in which it's called.
- This means a single method name might work in different ways depending on what arguments are passed to it.



Source: https://techvidvan.com/tutorials/method-overloading-and-overriding/

Method Overriding

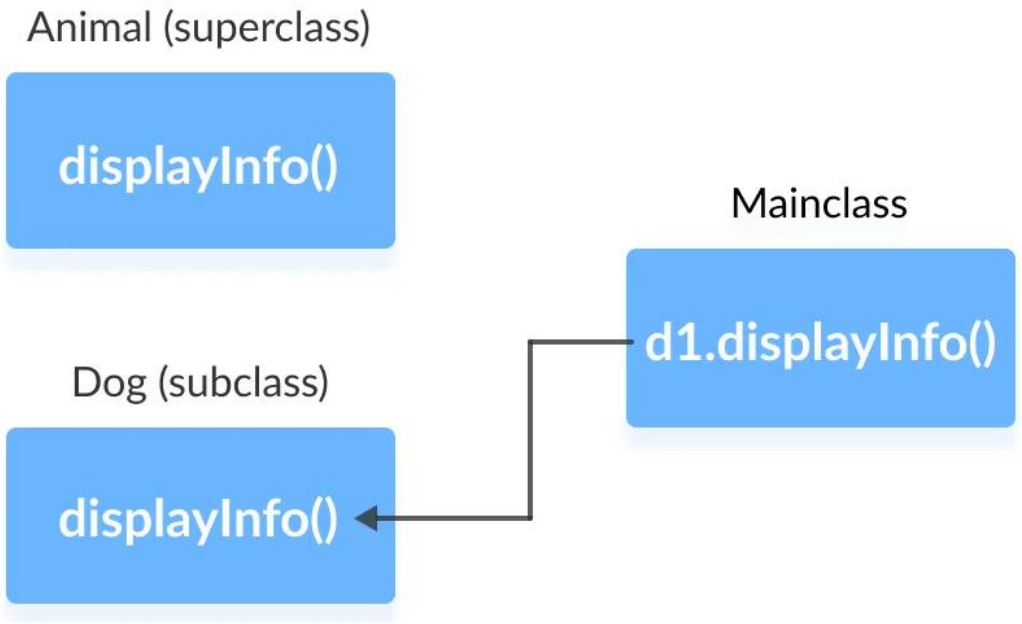
- The child class can override a method of its parent class.
- That allows a programmer to use one method in different ways depending on whether it's invoked by an object of the parent class or an object of the child class.



Source: https://techvidvan.com/tutorials/method-overloading-and-overriding/

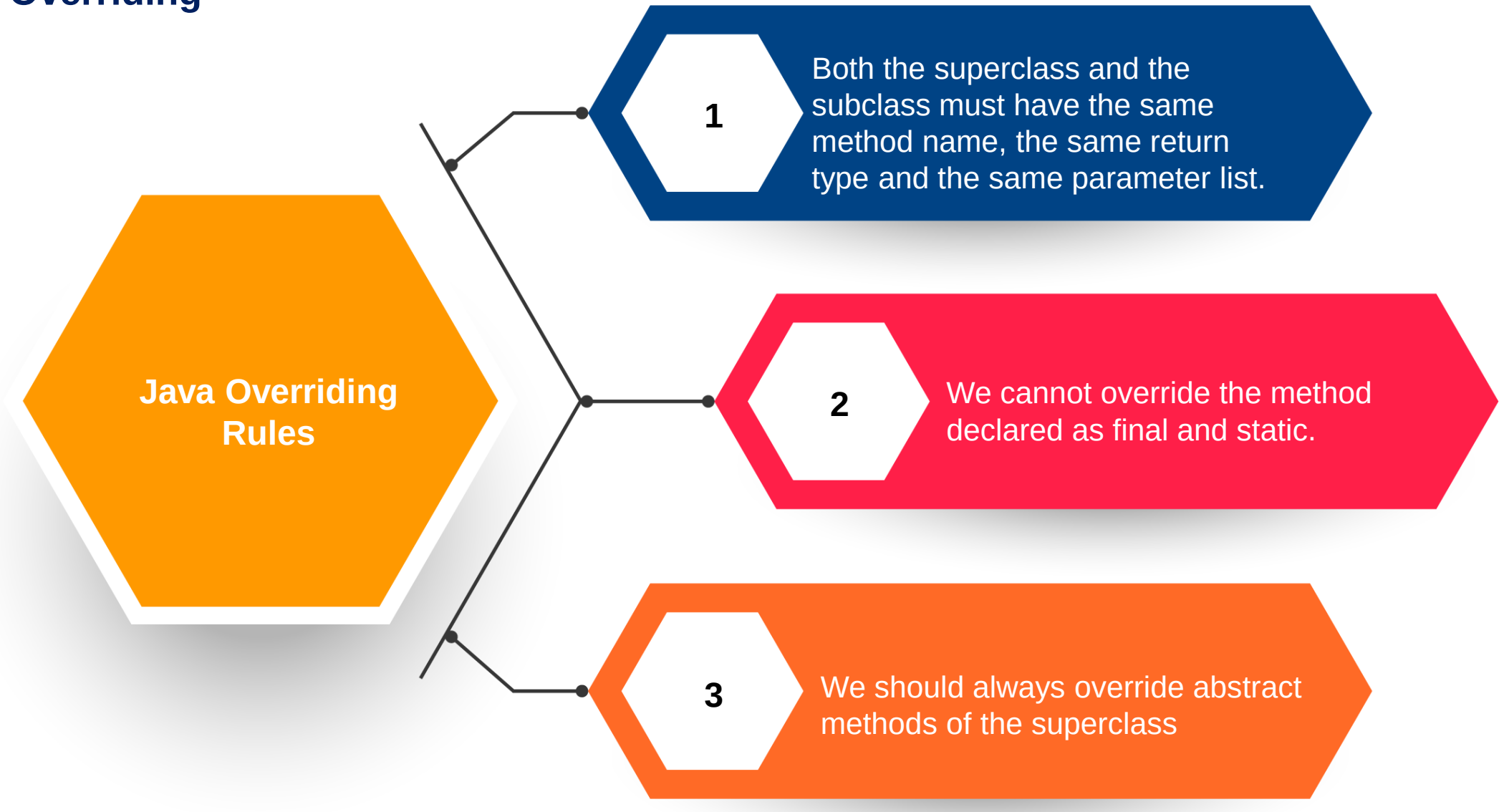
Method Overriding

- If the same method is defined in both the superclass and the subclass, then the method of the subclass class overrides the method of the superclass.
- This is known as method overriding.



Source: https://www.programiz.com/java-programming/method-overriding

Method Overriding



edunet foundation

Lab Exercise

Hands On - 10: Demonstrate polymorphism in java

Conclusion

Well done! You have completed this course and now you understand about:

- Java Object and Class
- Java Packages
- Methods
- Constructor
- Access Modifiers, and Non-Access Modifiers
- Abstraction
- Encapsulation
- Inheritance
- Polymorphism
- interfaces

Quiz

1. Which feature of OOPS derives the class from another class?

a) Inheritance

b) Data hiding

c) Encapsulation

d) Polymorphism

Answer: a
Inheritance

Quiz

2. Which among the following cannot be used for the concept of polymorphism?

a) Static member function

b) Constructor Overloading

c) Member function overloading

d) Global member function

Answer: a
Static member function

Quiz

3. Which of the following OOP concept binds the code and data together and keeps them secure from the outside world?

a) Polymorphism

b) Inheritance

c) Abstraction

d) Encapsulation

Answer: d
Encapsulation

Quiz

4. Which class cannot create its instance?

a) Parent class

b) Nested class

c) Anonymous class

d) Abstract class

Answer: d
Abstract class

Quiz

5. The principle of abstraction_____

a) is used to achieve OOPS.
b) is used to avoid duplication
c) Use abstraction at its minimum
d) is used to remove longer codes

Answer: b
is used to avoid duplication

References

- https://dzone.com/articles/is-java-still-relevant-in-2021-1
- https://stackify.com/oops-concepts-in-java
- https://www.javatpoint.com/java-oops-concepts
- https://www.geeksforgeeks.org/introduction-of-object-oriented-programming/
- https://www.w3schools.com/java/java_oop.asp
- https://www.youtube.com/watch?v=bSrm9RXwBaI&t=1657s

edunet
foundation

Thank You!