Polymorphism

- Refers to the concept of having the **same name**, but **different forms**.

Types of Polymorphism

- **Compile-time Polymorphism** (*Method Overloading*)
    - Includes *Method Hiding*.
- **Run-time Polymorphism** (*Method Overriding*)

Compile-time Polymorphism (Method Overloading)

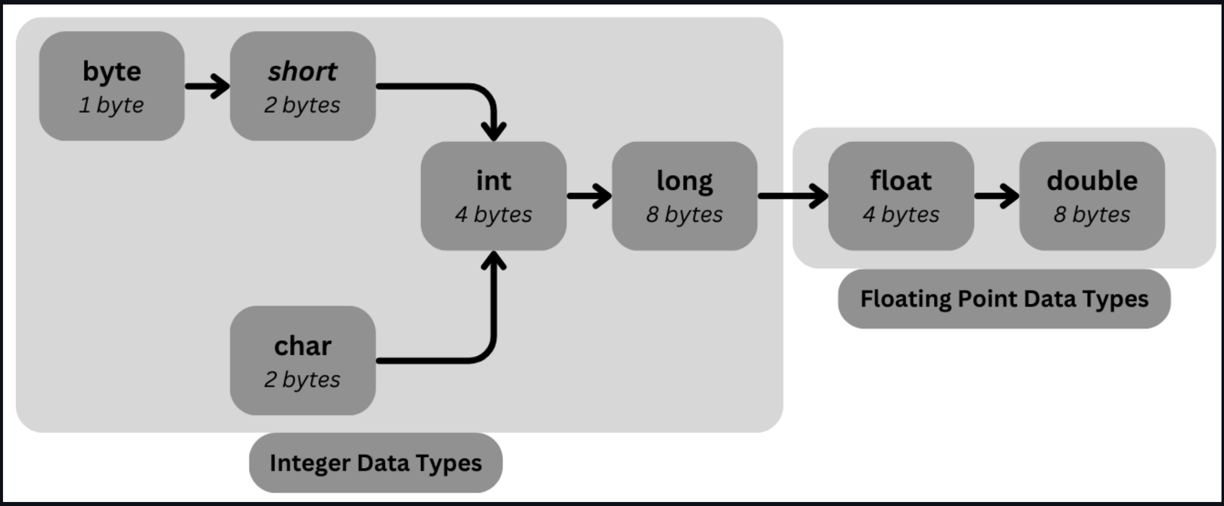
- Declaring *multiple* methods with the **same name** but with **different parameter lists** within the *same* `class`.

```java
class CompileTime {
    public void func(int a, int b) {/* ... */}
    public void func(int a) {/* ... */}
    public void func(int a, float b) {/* ... */}

    public static void main(String[] args) {
        // Pre-defined overloading
        System.out.println("1");
        System.out.println('1');
        System.out.println(1);
        System.out.println(true);

        // User-defined overloading
        CompileTime ob = new CompileTime();
        ob.func(2, 4);
        ob.func(5);
        ob.func(7, 9.6f);
    }
}
```

- **Upcasting** is possible in method overloading if the JVM does **not** find an **exact match**.
    - JVM chooses the **nearest** data type.



- When multiple matching parameters exist, **priority** is given to the **nearest** data type.

```java
class MatchingParameters {
    public void func(int a, double b) {
        System.out.println("double");
    }
    public void func(int a, float b) {
        System.out.println("float");
    }

    public static void main(String[] args) {
        MatchingParameters ob = new MatchingParameters();
        ob.func(2, 4.6d); // double
        ob.func(7, 9.6f); // float
    }
}
```

Non-primitive Parameters in Method Overloading

- If non-primitive arguments match multiple overloaded methods, the **JVM** selects based on **inheritance hierarchy**:
  - Preference is given to the **most specific** (child) class.
  - If no relationship exists, a **compilation error** occurs.

```java
class NonPrimitivePreference {
    public void func(Object o) {
        System.out.println("Object class");
    }
    public void func(String s) {
        System.out.println("String class");
    }
    public void func(NonPrimitivePreference npp) {
        System.out.println("NonPrimitivePreference class");
    }

    public static void main(String[] args) {
        NonPrimitivePreference ob = new NonPrimitivePreference();
        ob.func(new Object()); // Object class
        ob.func(new String()); // String class
        ob.func(null); // May result in ambiguity
    }
}
```

Run-time Polymorphism (Method Overriding)

- Providing a **different implementation** for a method inherited from a *parent class*.

Conditions

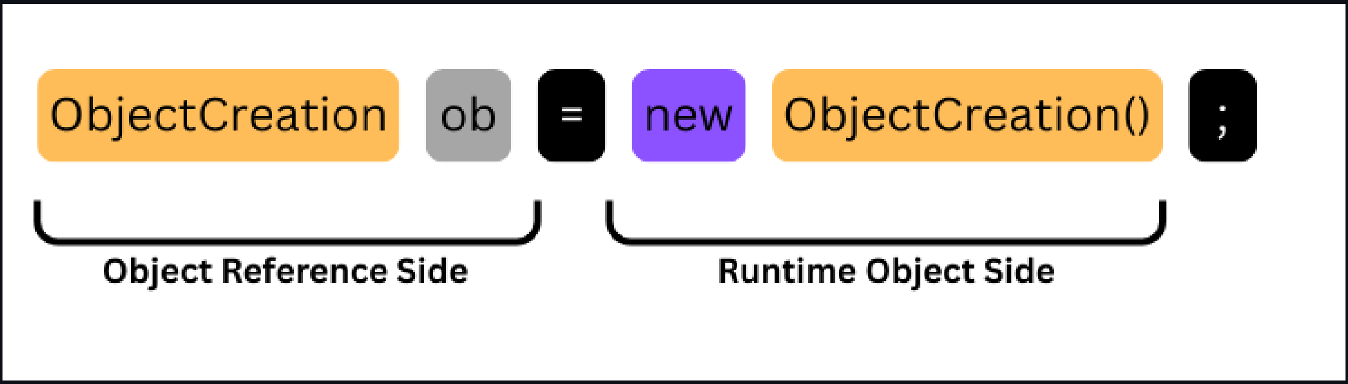
- At least **two** classes are required.
- A **Parent-Child** relationship must exist.
- Method signatures must be **identical**.
- Only **non-static** methods can be overridden.

```java
class A {
    public void func(String s) {
        System.out.println("Imp. of class A.");
    }
}

class B extends A {
    public void func(String s) {
        System.out.println("Imp. of class B.");
    }

    public static void main(String[] args) {
        B b = new B();
        b.func("argument"); // Imp. of class B.
    }
}
```

Method Resolution



- **Compiler** checks for method existence and signature.
- **JVM** determines actual method execution at run-time.
- Method execution is based on the **run-time object**.

Co-variant Return Type

- *Prior to Java 1.4*, return type changes in overridden methods were **not allowed**.
- *Post Java 1.4*, **co-variant return types** are allowed.

Conditions

- Only **non-primitive** return types are permitted.
- Return type in the **child class** must be a **subclass** of the return type in the parent class.

```java
class A {
    public A func(String s) {
        System.out.println("Imp. of class A.");
        return this;
    }
}

class B extends A {
    public B func(String s) {
        System.out.println("Imp. of class B.");
        return this;
    }
}

class C extends B {
    public static void main(String[] args) {
        C c = new C();
        B b = c.func("argument"); // Imp. of class B.
    }
}
```