Object Type Casting (Non-Primitive Type Casting)

- Object type casting involves assigning a **new reference type** to an **existing object**.
- This allows accessing the same object through a different type **without creating a new object**.
- Only the **reference type** changes — the **actual object remains unchanged**.

Checks for Object Type Casting

```java
Object ob = new String();
StringBuffer sb = (StringBuffer) ob;
     A       B          C         D
```

✅ Check 1 — Relationship Between `C` and `D`

- Evaluated at **compile-time**.
- There must be an **IS-A relationship** between `C` and `D` (i.e., one must be a subclass of the other).
- If not, the compiler raises an **incompatible types** error.

```java
Object ob = new String();
StringBuffer sb = (StringBuffer) ob; // Compile-time Error

String ob = new String();
StringBuffer sb = (StringBuffer) ob; // Compile-time Error
```

✅ Check 2 — Relationship Between `A` and `C`

- Also a **compile-time** check.
- `A` and `C` must either be the **same type**, or `A` must be a **superclass of C**.
- Otherwise, the compiler throws an **incompatible types** error.

```java
Object ob = new String();
StringBuffer sb = (StringBuffer) ob; // Compile-time Error

String ob = new String();
Object sb = (StringBuffer) ob; // Valid if cast is legal

Object ob = new String();
String sb = (StringBuffer) ob; // Compile-time Error
```

✅ Check 3 — Actual Object Type Validation

- Performed at **runtime** by the **JVM**.
- The actual object type of `D` must be compatible with `C`.
- If not, a `ClassCastException` will occur.

```java
Object ob = new String();
StringBuffer sb = (StringBuffer) ob; // Runtime Error:
ClassCastException

Object ob = new Integer(123);
Number n = (Number) ob; // Valid
```

Key Principles

- **No new object** is created during object type casting.
- The **existing object** is accessed using a new **reference type**.

```java
class Parent {
    void displayParent() {
        System.out.println("Parent method");
    }
}

class Child {
    void displayChild() {
        System.out.println("Child method");
    }
}
```

```java

public class TypeCastDemo {
    public static void main(String[] args) {
        Child child = new Child();
        child.displayChild();

        // Attempting to cast to unrelated type
        ((Parent) child).displayChild(); // Compile-time Error
    }
}
```

> **Explanation:**
>
> - `Child` does not extend `Parent`, so `(Parent) child` is invalid.
> - The compiler throws an error because `Parent` does not contain `displayChild()`.