Downcasting, Upcasting, and equals() Method in Java

Lecture 19

Exploring Type Casting and Object Comparison

Downcasting and Upcasting

What is upcasting?

Definition:

- Converting a subtype reference to a supertype reference.

Key Points:

- Happens automatically in Java.
- Ensures type compatibility for polymorphism.

```
Animal a = new Dog();  // Upcasting
a.speak();  // Calls Dog's overridden method
```

Benefits of Upcasting

- **Allows treating specific objects uniformly as supertype.**
- **Enables code flexibility and supports polymorphism.**

What is Downcasting?

Definition:

- Converting a supertype reference back to a subtype reference.

Key Points:

- Requires explicit casting.
- Used when specific behavior of the subtype is needed.

```
Animal a = new Dog();  // Upcasting

Dog d = (Dog) a;  // Downcasting

d.fetch();
```

Risks of Downcasting

ClassCastException:

- Occurs when casting between unrelated types.

Solution:

- Use instanceof to ensure safe downcasting.

```
if (a instanceof Dog) {
    Dog d = (Dog) a;
    d.fetch();
}
```

Generic Types, and Static & Dynamic Typing in Java

Exploring Type Safety and Flexibility in Programming

What are Generic Types?

Definition:

- Generics enable types (classes and methods) to be parameterized, ensuring type safety at **compile time**.

Purpose:

- Avoids ClassCastException.
- Eliminates the need for casting while working with collections.

```
List<String> list = new ArrayList<>();

list.add("Hello");

String item = list.get(0);  // No casting needed
```

Advantages of Generic Types

Type Safety: Ensures type correctness at compile time.

Code Reusability: Enables the use of a single method or class for different data types.

Eliminates Casting: Simplifies code readability and reduces errors.

Real-World Example

Scenario:

- Using generics for a data repository.

```
public class Repository<T> {
    private List<T> items = new ArrayList<>();
    public void add(T item) {
        items.add(item);
    }
    public T get(int index) {
        return items.get(index);
    }
}
```

Static & Dynamic Typing

What is Static Typing?

- **Definition:**
  - Type checking is performed at compile time.
- **Characteristics:**
  - Errors are caught early.
  - Strong type enforcement.
- **Examples:**
  - Java, C, C++.

```
int number = 5;  // Type is fixed
```

What is Dynamic Typing?

Definition:

- Type checking is performed at runtime.

Characteristics:

- More flexible, but errors occur at runtime.

Examples:

- Python, JavaScript.

```
number = 5  # Type can change later
number = "Five"
```

Real-World Scenarios

Static Typing Example: Banking applications requiring high reliability and error detection.

Dynamic Typing Example: Prototyping or scripting for rapid development.