Polymorphism in Array / Collection of Objects

1️⃣ Concept

- Parent class type reference দিয়ে child class objects handle করা যায়।
- অনেকগুলো child object একই parent type reference দিয়ে array বা collection তে store করা যায়।
- Dynamic method dispatch এবং runtime polymorphism সহজ হয়।

2️⃣ Using Array of Objects

```
abstract class Animal {
    abstract void sound();
}

class Dog extends Animal {
    void sound() {
        System.out.println("Dog barks");
    }
}

class Cat extends Animal {
    void sound() {
        System.out.println("Cat meows");
    }
}

public class TestArray {
    public static void main(String[] args) {
        Animal[] animals = new Animal[2];
        animals[0] = new Dog();
        animals[1] = new Cat();

        for (Animal a : animals) {
            a.sound();
        }
    }
}
```

Output:

```
Dog barks
Cat meows
```

3️⃣ Using Collection (ArrayList) of Objects

```java
import java.util.ArrayList;

abstract class Animal {
    abstract void sound();
}

class Dog extends Animal {
    void sound() {
        System.out.println("Dog barks");
    }
}

class Cat extends Animal {
    void sound() {
        System.out.println("Cat meows");
    }
}

public class TestCollection {
    public static void main(String[] args) {
        ArrayList<Animal> animals = new ArrayList<>();
        animals.add(new Dog());
        animals.add(new Cat());

        for (Animal a : animals) {
            a.sound();
        }
    }
}
```

Output:

```
Dog barks
Cat meows
```

4️⃣ Key Points

1. Parent type reference দিয়ে child objects array বা collection এ store করা যায়।
2. Runtime-এ proper child method call হয়।
3. Dynamic method dispatch এবং runtime polymorphism এর প্রধান ব্যবহার।
4. Collection (ArrayList, Vector, etc.) ব্যবহার করলে size flexible থাকে।

💡 Shortcut: Parent type reference → multiple child objects → runtime method dispatch.