# 05_3 Copy Constructor

Object-Oriented Programming

# Copy Constructor

- Constructor with a single argument of the same type as the class
- Should create an object that is a separate, independent object
- Values of instance variables are the same as original

# Copy Constructor Example

```
public class Person {
    private String name;
    private int age;

    // Copy Constructor
    public Person(Person person) {
        this.name = person.name;
        this.age = person.age;
    }
```

```
    // Original Person object
    Person original = new Person("John Doe", 30);

    // Using copy constructor
    Person copy = new Person(original);
```

# Example: Person.java (1/4)

```
public class Person {
    private String name;
    private int age;

    // default constructor
    public Person() { }

    // constructor
    public Person(String name, int age) {
        this.name = name;
        this.age = age;
    }

    // Copy Constructor
    public Person(Person person) {
        this.name = person.name;
        this.age = person.age;
    }
```

# Example: Person.java (2/4)

```
    // Accessors and Mutators
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    // toString
    public String toString() {
        return "Person{name='" + name + "', age=" + age + '}';
    }
```

# Example: Person.java (3/4)

```
    public static void main(String[] args) {

        Person original = new Person("John Doe", 30);
        System.out.println("Original: " + original);

        // Using copy constructor
        Person copy = new Person(original);
        System.out.println("Copy: " + copy);

        // Test the equality between original and copy objects
        System.out.println("Are the objects equal? " + original.equals(copy));

        // Test the privacy leak
        copy.setName("Jane Doe");
        copy.setAge(25);
        System.out.println("Modified Copy: " + copy);
        System.out.println("Original after modifying copy: " + original);
    }
}
```

# Example: Person.java (4/4)

```
OUTPUT:

Original: Person{name='John Doe', age=30}
Copy: Person{name='John Doe', age=30}
Are the objects equal? false  // because the two objects' reference are different
Modified Copy: Person{name='Jane Doe', age=25}
Original after modifying copy: Person{name='John Doe', age=30}
// privacy leak prevented
```

# Example: Person2.java (1/7)

```
public class Person2 {
    private String name;
    private int age;
    private Address address; // Person2 has an Address: class variable

    // class variable
    private static String country = "Unknown";

    // default constructor
    public Person2() { }

    // constructor
    public Person2(String name, int age, Address address) {
        this.name = name;
        this.age = age;
        this.address = address;
    }
```

# Example: Person2.java (2/7)

```
    // Shallow Copy Constructor
    public Person2(Person2 person) {
        this.name = person.name;
        this.age = person.age;
        this.address = person.address; // Shallow copy (just copy the reference)
    }

    // Deep Copy Constructor
    public Person2 deepCopy(Person2 person) {
        return new Person2(person.name, person.age, new Address(person.address));
    }

    // Accessors and Mutators
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
```

# Example: Person2.java (3/7)

```
    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public Address getAddress() {
        return address;
    }

    public void setAddress(Address address) {
        this.address = address;
    }

    public static String getCountry() {
        return country;
    }
```

# Example: Person2.java (4/7)

```
    public static void setCountry(String country) {
        Person2.country = country;
    }

    // toString
    public String toString() {
        return "Person2{name='" + name + "', age=" + age + ", address=" + address
                + ", country=" + country + '}';
    }

    public static void main(String[] args) {
        // original object creation
        Address address = new Address("Seoul", "1234 Street");
        Person2 original = new Person2("John Doe", 30, address);
        System.out.println("Original: " + original);


        // shallow copy
        Person2 shallowCopy = new Person2(original);
        System.out.println("Shallow Copy: " + shallowCopy);
```

# Example: Person2.java (5/7)

```
// Using Deep Copy Constructor
Person2 deepCopy = original.deepCopy(original);
System.out.println("Deep Copy: " + deepCopy);


// Test the equality of original and copy
System.out.println("Are the shallow copy objects equal?"+ original.equals(shallowCopy));
System.out.println("Are the deep copy objects equal? " + original.equals(deepCopy));


// Test the privacy leak
shallowCopy.setName("Jane Doe");
shallowCopy.setAge(25);
shallowCopy.getAddress().setCity("Busan");
shallowCopy.getAddress().setStreet("5678 Avenue");
System.out.println("Modified Shallow Copy: " + shallowCopy);
System.out.println("Original after modifying shallow copy: " + original);
```

# Example: Person2.java (6/7)

```
        deepCopy.setName("Alice Smith");
        deepCopy.setAge(28);
        deepCopy.getAddress().setCity("Incheon");
        deepCopy.getAddress().setStreet("91011 Boulevard");
        System.out.println("Modified Deep Copy: " + deepCopy);
        System.out.println("Original after modifying deep copy: " + original);

        // Test the class variable change
        Person2.setCountry("Korea");
        System.out.println("Original after changing country: " + original);
        System.out.println("Shallow Copy after changing country: " + shallowCopy);
        System.out.println("Deep Copy after changing country: " + deepCopy);
    }
}
```

# Example: Person2.java (7/7)

OUTPUT:

Original: Person2{name='John Doe', age=30, address=Address{city='Seoul', street='1234 Street'}, country=Unknown}
Shallow Copy: Person2{name='John Doe', age=30, address=Address{city='Seoul', street='1234 Street'}, country=Unknown}
Deep Copy: Person2{name='John Doe', age=30, address=Address{city='Seoul', street='1234 Street'}, country=Unknown}
Are the shallow copy objects equal? false
Are the deep copy objects equal? false
Modified Shallow Copy: Person2{name='Jane Doe', age=25, address=Address{city='Busan', street='5678 Avenue'}, country=Unknown}
Original after modifying shallow copy: Person2{name='John Doe', age=30, address=Address{city='Busan', street='5678 Avenue'}, country=Unknown}
Modified Deep Copy: Person2{name='Alice Smith', age=28, address=Address{city='Incheon', street='91011 Boulevard'}, country=Unknown}
Original after modifying deep copy: Person2{name='John Doe', age=30, address=Address{city='Busan', street='5678 Avenue'}, country=Unknown}
Original after changing country: Person2{name='John Doe', age=30, address=Address{city='Busan', street='5678 Avenue'}, country=Korea}
Shallow Copy after changing country: Person2{name='Jane Doe', age=25, address=Address{city='Busan', street='5678 Avenue'}, country=Korea}
Deep Copy after changing country: Person2{name='Alice Smith', age=28, address=Address{city='Incheon', street='91011 Boulevard'}, country=Korea}