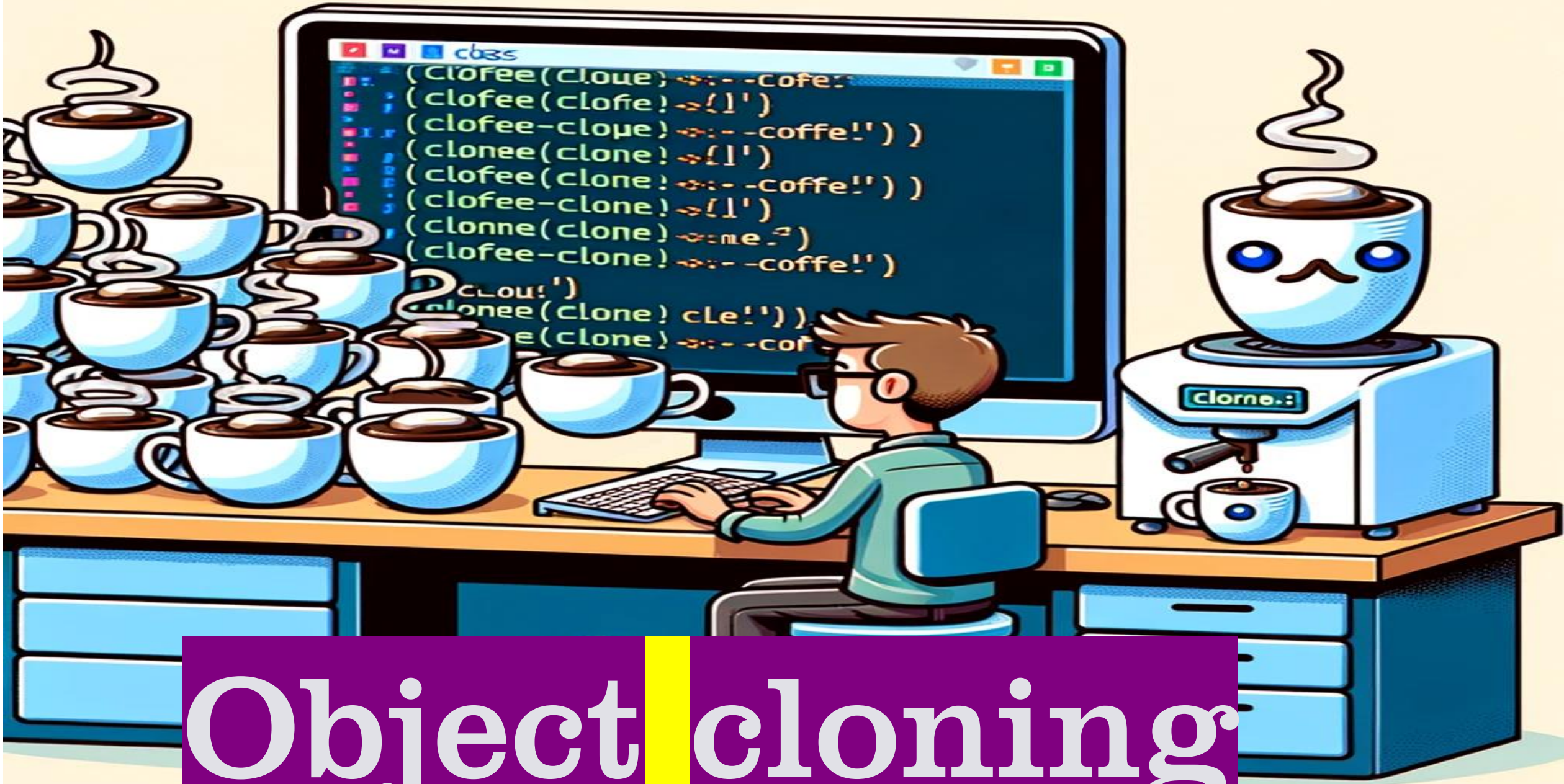

Object cloning

The object cloning is a way to create exact copy of an object. The clone() method of Object class is used to clone an object.

The java.lang.Cloneable interface must be implemented by the class whose object clone we want to create. If we don't implement Cloneable interface, clone() method generates CloneNotSupportedException.

Syntax of the clone() method is as follows:

protected Object clone() throws CloneNotSupportedException

Object cloning refers to creation of exact copy of an object. It creates a new instance of the class of current object and initializes all its fields with exactly the contents of the corresponding fields of this object. In Java, there is no operator to create copy of an object. UnlikeC++, in Java, if we use assignment operator then it will create a copy of reference variable and not the object.

How to Implement Cloning

Java provides a built-in mechanism for object cloning through the **Cloneable** interface and the **clone()** method from the **Object** class. Here are the steps involved:

1.Implement the Cloneable Interface: This interface is a marker interface (it contains no methods) that indicates that a class is legally cloneable.

2.Override the clone() Method: You need to override the protected **clone()** method from the **Object** class in your class. This method needs to call **super.clone()** to create a shallow copy of the object.

3.Handle the CloneNotSupportedException: Since **clone()** can throw a **CloneNotSupportedException**, you need to handle this exception, either by throwing it or by catching it within the method.

```
// Define a class Car that implements the Cloneable interface
class Car implements Cloneable {
    private String make;
    private int modelYear;

    // Constructor to initialize the Car object
    public Car(String make, int modelYear) {
        this.make = make;
        this.modelYear = modelYear;
    }

    // Getter methods
    public String getMake() {
        return make;
    }

    public int getModelYear() {
        return modelYear;
    }
```

```
    // Implement the clone method to make this object cloneable
    @Override
    protected Object clone() throws CloneNotSupportedException {
        return super.clone();
    }

    // Method to display the attributes of the car
    public void displayInfo() {
        System.out.println("Car make: " + make + ", Model Year: " + modelYear);
    }
}

public class Main {
    public static void main(String[] args) {
        try {
            // Create an instance of Car
            Car originalCar = new Car("Toyota", 2021);

            // Clone the originalCar
            Car clonedCar = (Car) originalCar.clone();
```

```
// Display information of both original and cloned car
        System.out.println("Original Car:");
        originalCar.displayInfo();
        System.out.println("Cloned Car:");
        clonedCar.displayInfo();

    } catch (CloneNotSupportedException e) {
        System.out.println("Cloning not supported");
    }
  }
}
```

Output

```
java -cp /tmp/8ZpUBuas1U/Main
Original Car:
Car make: Toyota, Model Year: 2021
Cloned Car:
Car make: Toyota, Model Year: 2021

=== Code Execution Successful ===
```

Shallow Object Cloning:

Definition: Shallow cloning creates a new object and copies all fields of the original object to the new object. However, if the fields themselves are reference types (objects), only the references to those objects are copied, not the objects themselves. This means that the new object shares references to the same objects as the original object.

Behavior: Changes made to mutable objects within the original object will affect the cloned object since they are referring to the same objects in memory.

```
class Person implements Cloneable {
   private String name;
   private Address address;
   public Person(String name, Address address) {
      this.name = name;
      this.address = address;
   }
   @Override
   protected Object clone() throws CloneNotSupportedException {
      return super.clone();
   } }
class Address {
   private String city;
   public Address(String city) {
      this.city = city;
   } }
public class Main {
   public static void main(String[] args) throws CloneNotSupportedException {
      Address address = new Address("New York");
      Person originalPerson = new Person("John", address);
      Person clonedPerson = (Person) originalPerson.clone();
      // Change city in original object
      originalPerson.getAddress().setCity("Los Angeles");
      System.out.println("Cloned Person's City: " + clonedPerson.getAddress().getCity()); // Output: Los Angeles
   } }
```

Deep Object Cloning:

Definition: Deep cloning creates a new object and recursively copies all fields of the original object to the new object, including any objects referenced by the fields. This ensures that the new object has its own separate copies of all referenced objects.

Behavior: Changes made to mutable objects within the original object will not affect the cloned object since they refer to different objects in memory.

```
class Person implements Cloneable {
   private String name;
   private Address address;
   public Person(String name, Address address) {
      this.name = name;
      this.address = address;}
   @Override
   protected Object clone() throws CloneNotSupportedException {
      Person cloned = (Person) super.clone();
      cloned.address = (Address) address.clone(); // Deep copy of Address object
      return cloned;   } }
class Address implements Cloneable {
   private String city;
   public Address(String city) {
      this.city = city;}
   @Override
   protected Object clone() throws CloneNotSupportedException {
      return super.clone();    } }
public class Main {
   public static void main(String[] args) throws CloneNotSupportedException {
      Address address = new Address("New York");
      Person originalPerson = new Person("John", address);
      Person clonedPerson = (Person) originalPerson.clone();
      // Change city in original object
      originalPerson.getAddress().setCity("Los Angeles");
      System.out.println("Cloned Person's City: " + clonedPerson.getAddress().getCity()); // Output: New York }}
```