Control Abstraction in Java with Examples

Difficulty Level : Medium Last Updated : 03 Mar, 2021

Our aim is to understand and implement Control Abstraction in Java. Before jumping right into control abstraction, let us understand what is abstraction.

Abstraction: To put it in simple terms, abstraction is anything but displaying only the essential features of a system to a user without getting into its details. For example, a car and its functions are described to the buyer and the driver also learns how to drive using the steering wheel and the accelerators but the inside mechanisms of the engine are not displayed to the buyer. To read more about Abstraction, refer here.

In abstraction, there are two types: Data abstraction and Control abstraction.

Data abstraction, in short means creating complex data types but giving out only the essentials operations.

Control Abstraction: This refers to the software part of abstraction wherein the program is simplified and unnecessary execution details are removed.

Here are the ***main points about control abstraction:***

- Control Abstraction follows the basic rule of DRY code which means Don't Repeat Yourself and using functions in a program is the best example of control abstraction.
- Control Abstraction can be used to build new functionalities and combines control statements into a single unit.
- It is a fundamental feature of all higher-level languages and not just java.
- Higher-order functions, closures, and lambdas are few preconditions for control abstraction.
- Highlights more on how a particular functionality can be achieved rather than describing each detail.
- Forms the main unit of structured programming.

A simple algorithm of control flow:

- The resource is obtained first
- Then, the block is executed.
- As soon as control leaves the block, the resource is closed

Example:

```
// Abstract class
abstract class Vehicle {
    // Abstract method (does not have a body)
    public abstract void VehicleSound();
    // Regular method
    public void honk() { System.out.println("honk honk"); }
}

// Subclass (inherit from Vehicle)
class Car extends Vehicle {
    public void VehicleSound()
    {
        // The body of VehicleSound() is provided here
        System.out.println("kon kon");
    }
}

class Main {
    public static void main(String[] args)
    {
        // Create a Car object
        Car myCar = new Car();
        myCar.VehicleSound();
        myCar.honk();
    }
}
```

Output

```
kon kon
honk honk
```

The greatest **advantage** of control abstraction is that it makes code a lot cleaner and also more secure.