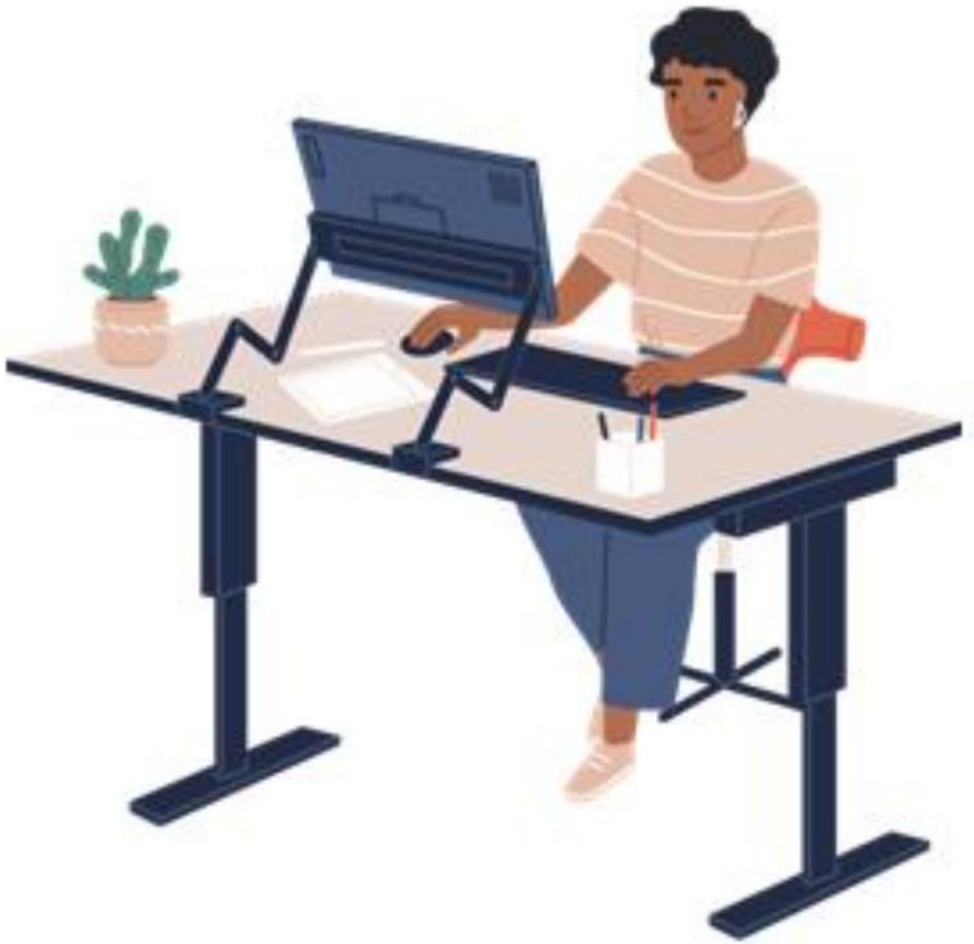
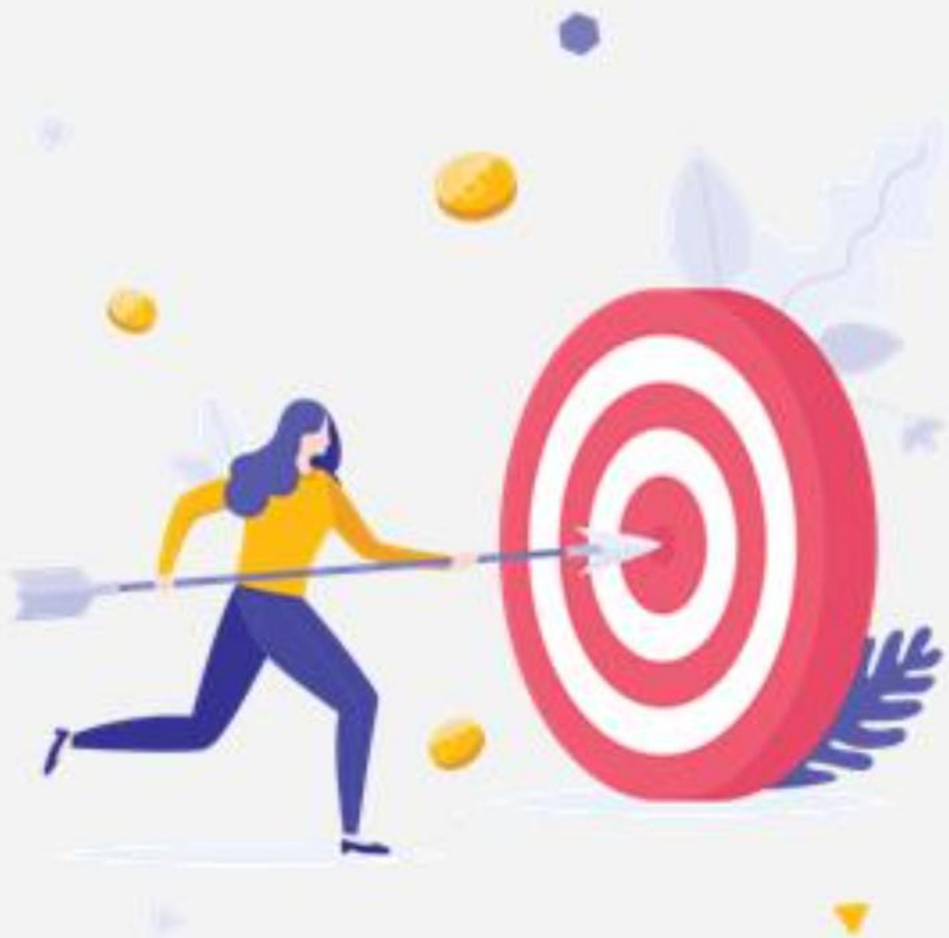
Learning Consolidation

Introduction to Encapsulation and Data Abstraction





Learning Objectives

- Explore user-defined packages
- Explain encapsulation
- Use access specifiers
- Explore naming conventions for getters/setters

User-Defined Packages

```java
package com.stackroute.employees;

public class Employee {
    int employeeCode;
    String employeeName;
    int age;
    String dob;
    String address;
    double salary;
}
```

The class `Employee` belongs to the `com.stackroute.employees` package.

- User-defined packages can be created by the user to organize classes.
- IDEs provide a structure to the Java project and automatically create a package structure.
- The first line of the class defines the package in which the class is present.
- `package` is a keyword used to specify which package the class belongs to.
- The package name must be lowercase.

Package Naming Conventions

- The packages we define must follow some naming conventions.

```
<com/org/gov/edu>.<xxx>.<yyy>
```

Package Name	Meaning
com	Symbolizes the type of organization for which applications are built Gov: for government organization Edu: for educational institution.
xxx	Name of the organization
yyy	Further categorization

Advantages of Packages

- A package is used to categorize the classes and interfaces to facilitate easy maintenance.
- Packages provide access protection.
- They eliminate naming collisions.

What Is Encapsulation?

- The wrapping of data and the code (i.e., the methods that work on the data are called **encapsulation**.)
- The data consists of the variables declared in the class.
- A class consists of variables and methods that work on those variables.

```java
public class Employee {
    int employeeCode;
    String employeeName;
    int age;
    String dob;
    String address;
    double salary;
    public Employee(int employeeCode, String employeeName, int age,
                    String dob, String address, double salary) {
        this.employeeCode = employeeCode;
        this.employeeName = employeeName;
        this.age = age;
        this.dob = dob;
        this.address = address;
        this.salary = salary;
    }
    void displayEmployeeDetails(){
        System.out.println(employeeName+"::"+employeeCode+"::"+salary);
        System.out.println(dob+"::"+age+"::"+address);
    }

    double calculateAnnualSalary(){
        return salary * 12;
    }

    double calculateSalaryHike(float payHikePercentage){
        this.salary = salary + (salary*payHikePercentage/100);
        return salary;
    }
}
```

variables

methods

Encapsulation in Java

- Encapsulation is achieved in Java using:
  - Classes: Classes wrap the variables and methods into one single unit accessed only by creating an object.
  - Access specifiers: Access specifiers restrict access to the variables in a class at multiple levels.
  - Setter and getter methods: These methods prevent variables from misuse or unwanted changes by other objects.

Access Specifiers

- The table below specifies how the access specifiers facilitate-a class, variable, or method to be used in classes within the same package or outside the package.

	Default	Private	Protected	Public
Same class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Naming Conventions for Getter/Setter Methods

- Getter and setter method names are composed of the words *get* or *set*, respectively, plus the property name with the first character of each word capitalized.
- A regular getter method has no parameters but returns a value of the property's type.
- A setter method has a single parameter of the property's type and has a void return type.
- Consider a class `Customer` with the attribute `customer name`. The setter and getter must be as follows:

```
public class Customer {

    private String customerName;
```

```
public String getCustomerName() {
    return customerName;
}

public void setCustomerName(String customerName) {
    this.customerName = customerName;
}
```

Naming Conventions for Getter/Setter Methods

- If a variable is declared as `boolean`, which represents if the customer is a premium member, the getter and setter must be as follows:

```
private boolean isPremiumMember;
```

- Note that the getter method does not have a "get" prefixed to the variable name.
- The setter method is prefixed with "set," as per convention.

```
public boolean isPremiumMember() {
    return isPremiumMember;
}

public void setPremiumMember(boolean premiumMember) {
    isPremiumMember = premiumMember;
}
```

Constructors vs. Getter/Setter

Constructors	Getter/Setter
Constructors are used to create an object of the class by initializing values of the instance variables.	The getter method is used to retrieve a single value of the instance variable, and setter is used to set or assign a single value of the instance variable.
Constructors do not create an encapsulated class.	A well-encapsulated class can be created by generating accessor and mutator methods or getter and setter methods for the variables.