Name of the Course : LEARNING JAVA 9 – MODULAR PROGRAMMING Using Inheritance

Level : High

Tool Stack : Java 9

Problem Statement : Inheritance is a powerful yet overused and misused mechanism.

Simply put, with inheritance, a base class (a.k.a. base type) defines the state and behavior common for a given type and lets the subclasses (a.k.a. subtypes) provide specialized versions of that state and behavior.

Step 1 : let's create base class Person that defines the common fields and methods for a person, while the subclasses Waitress and Actress provide additional, fine-grained method implementations.

**Code:**

**package** com.iiht.java9.javainheritance;

**public** **class** Person {

**private** **final** String name;

**private** **final** String email;

**private** **final** **int** age;

**public** Person(String name, String email, **int** age) {

**this**.name = name;

**this**.email = email;

**this**.age = age;

}

**public** String getName() {

**return** name;

}

**public** String getEmail() {

**return** email;

}

**public** **int** getAge() {

**return** age;

}

@Override

**public** String toString() {

**return** "Person{" + "name=" + name + ", email=" + email + ", age=" + age + "}";

}

}

Step 2 : And these are the subclasses:

**Code:**

**package** com.iiht.java9.javainheritance;

**public** **class** Actress **extends** Person {

**public** Actress(String name, String email, **int** age) {

**super**(name, email, age);

}

**public** String readScript(String movie) {

**return** "Reading the script of " + movie;

}

**public** String performRole() {

**return** "Performing a role";

}

}

**package** com.iiht.java9.javainheritance;

**public** **class** Waitress **extends** Person {

**public** Waitress(String name, String email, **int** age) {

**super**(name, email, age);

}

**public** String serveStarter(String starter) {

**return** "Serving a " + starter;

}

**public** String serveMainCourse(String mainCourse) {

**return** "Serving a " + mainCourse;

}

**public** String serveDessert(String dessert) {

**return** "Serving a " + dessert;

}

}

Step 3 : Develop The test class to test the functionality of application developed

**Code:**

**package com.iiht.java9.javainheritance;**

**import static org.junit.jupiter.api.Assertions.\*;**

**import org.junit.jupiter.api.AfterAll;**

**import org.junit.jupiter.api.AfterEach;**

**import org.junit.jupiter.api.Test;**

**import com.iiht.java9.javainheritance.Actress;**

**import com.iiht.java9.javainheritance.Person;**

**import com.iiht.java9.javainheritance.Waitress;**

**import java.util.Optional;**

**import static org.assertj.core.api.Assertions.assertThat;**

**class JavaInheritanceTest {**

**@AfterAll**

**static void tearDownAfterClass() throws Exception {**

**}**

**@AfterEach**

**void tearDown() throws Exception {**

**}**

**@Test**

**public void givenWaitressInstance\_whenCheckedType\_thenIsInstanceOfPerson() {**

**assertThat(new Waitress("Mary", "mary@domain.com", 22))**

**.isInstanceOf(Person.class);**

**}**

**@Test**

**public void givenActressInstance\_whenCheckedType\_thenIsInstanceOfPerson() {**

**assertThat(new Actress("Susan", "susan@domain.com", 30))**

**.isInstanceOf(Person.class);**

**}**

**}**

Learning outcome: Participant could able to learn how to use class Inheritance relationship using Java 9.