Институт приборостроения, автоматизации и информационных технологий

Кафедра информационных систем и цифровых технологий

Дисциплина «Качество и тестирование программного обеспечения»

Отчет к лабораторной работе № 8

«Объектно-ориентированное тестирование»

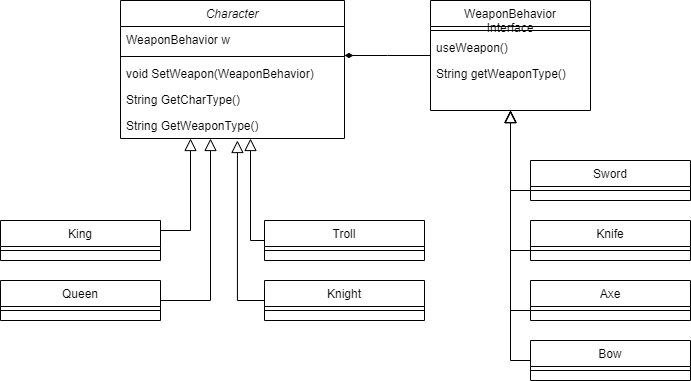
Выполнил:

Василения Иван Валерьевич

Принял:

Олькина Елена Викторовна

Орёл, 2025г



Листинг тестов:

package lab8;

import java.io.Console;

import lab8.\*;

import lab8.chars.\*;

import lab8.weapons.\*;

import org.junit.jupiter.api.Test;

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

class AppTest {

@Test void TypeTests() {

Sword sword = new Sword();

Knife knife = new Knife();

Axe axe = new Axe();

Bow bow = new Bow();

assertEquals(sword.getWeaponType(), "Sword");

assertEquals(knife.getWeaponType(), "Knife");

assertEquals(axe.getWeaponType(), "Axe");

assertEquals(bow.getWeaponType(), "Bow");

King k = new King(sword);

Queen q = new Queen(knife);

Troll t = new Troll(axe);

Knight n = new Knight(bow);

assertEquals(k.getCharType(), "King");

assertEquals(q.getCharType(), "Queen");

assertEquals(t.getCharType(), "Troll");

assertEquals(n.getCharType(), "Knight");

}

@Test void WeaponTypeTests() {

King k = new King(new Sword());

Queen q = new Queen(new Knife());

Troll t = new Troll(new Axe());

Knight n = new Knight(new Bow());

assertEquals(k.getWeaponType(), "Sword");

assertEquals(q.getWeaponType(), "Knife");

assertEquals(t.getWeaponType(), "Axe");

assertEquals(n.getWeaponType(), "Bow");

}

@Test void ChangeWeaponTests() {

King k = new King(new Sword());

Queen q = new Queen(new Knife());

Troll t = new Troll(new Axe());

Knight n = new Knight(new Bow());

k.setWeapon(new Knife());

q.setWeapon(new Sword());

t.setWeapon(new Bow());

n.setWeapon(new Axe());

assertEquals(k.getWeaponType(), "Knife");

assertEquals(q.getWeaponType(), "Sword");

assertEquals(t.getWeaponType(), "Bow");

assertEquals(n.getWeaponType(), "Axe");

}

@Test void testStrategyCompatibility() {

King k = new King(new Sword());

Queen q = new Queen(new Knife());

k.setWeapon(q.getWeapon());

assertEquals(k.getWeaponType(), "Knife");

}

@Test void childTest() {

Character someChar1 = new King(new Sword());

assertEquals(someChar1.getWeaponType(), "Sword");

assertEquals(someChar1.getCharType(), "King");

Character someChar2 = new Queen(new Knife());

assertEquals(someChar2.getWeaponType(), "Knife");

assertEquals(someChar2.getCharType(), "Queen");

}

@Test

void kingTest() {

King k = new King(new Sword());

assertThrows(

java.lang.IllegalAccessException.class,

() -> k.getClass().getSuperclass().getDeclaredField("weaponBehavior").get(k)

);

}

@Test

void queenTest() {

Queen q = new Queen(new Knife());

assertThrows(

java.lang.IllegalAccessException.class,

() -> q.getClass().getSuperclass().getDeclaredField("weaponBehavior").get(q)

);

}

@Test

void knightTest() {

Knight n = new Knight(new Bow());

assertThrows(

java.lang.IllegalAccessException.class,

() -> n.getClass().getSuperclass().getDeclaredField("weaponBehavior").get(n)

);

}

@Test void trollTest() {

Troll t = new Troll(new Axe());

assertThrows(

java.lang.IllegalAccessException.class,

() -> t.getClass().getSuperclass().getDeclaredField("weaponBehavior").get(t)

);

}

}