Phase-1 Practice Project: Assisted Practice

1.Writing a program in Java to implement implicit and explicit type casting.

Code and output:

**public** **class** TypeCastingExample {

**public** **static** **void** main(String[] args) {

**int** a=10;

**float** b=a;

System.***out***.println("Implicit type casting: int to float");

System.***out***.println("a = " + a + ", b = " + b);

**double** c = 15.75;

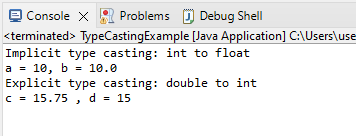
**int** d = (**int**) c;

System.***out***.println("Explicit type casting: double to int");

System.***out***.println("c = " + c + " , d = " + d);

}

}



2.Writing a program in Java to implement access modifiers.

**public** **class** AccessModifiersDemo {

// Private instance variables

**private** String privateVar = "This is a private variable";

// Protected instance variables

**protected** String protectedVar = "This is a protected variable";

// Public instance variables

**public** String publicVar = "This is a public variable";

// Private method

**private** **void** privateMethod() {

System.***out***.println("This is a private method");

}

// Protected method

**protected** **void** protectedMethod() {

System.***out***.println("This is a protected method");

}

// Public method

**public** **void** publicMethod() {

System.***out***.println("This is a public method");

}

**public** **static** **void** main(String[] args) {

// Create an object of the AccessModifiersDemo class

AccessModifiersDemo obj = **new** AccessModifiersDemo();

// Access instance variables and methods

System.***out***.println("Accessing instance variables and methods:");

System.***out***.println(obj.privateVar);

System.***out***.println(obj.protectedVar);

System.***out***.println(obj.publicVar);

obj.privateMethod();

obj.protectedMethod();

obj.publicMethod();

// Access instance variables and methods from a subclass

System.***out***.println("\nAccessing instance variables and methods from a subclass:");

Subclass obj2 = **new** Subclass();

obj2.accessProtectedVar();

obj2.accessProtectedMethod();

}

}

// Subclass of AccessModifiersDemo

**class** Subclass **extends** AccessModifiersDemo {

// Method to access protected variable

**public** **void** accessProtectedVar() {

System.***out***.println("Accessing protected variable from subclass: " + protectedVar);

}

// Method to access protected method

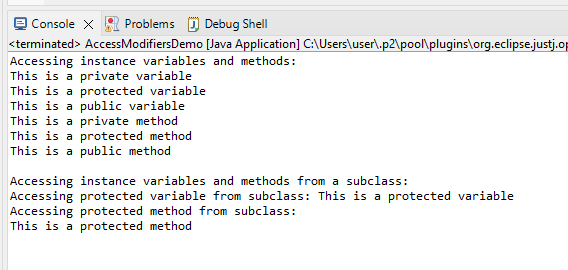
**public** **void** accessProtectedMethod() {

System.***out***.println("Accessing protected method from subclass:");

protectedMethod();

}

}



3. Writing a program in Java to verify implementations of methods and ways of calling a method

**public** **class** MethodVerification {

**public** **static** **int** add(**int** a, **int** b) {

**return** a + b;

}

**public** **static** String concatenate(String s1, String s2) {

**return** s1 + s2;

}

**public** **static** **void** main(String[] args) {

**int** sum = *add*(5, 7);

System.***out***.println("Sum of 5 and 7 is: " + sum);

String result = *concatenate*("Hello", "World");

System.***out***.println("Concatenated string is: " + result);

s

MethodVerification obj = **new** MethodVerification();

obj.sayHello();

MethodVerification.*sayBye*();

**int**[] numbers = {2, 4, 6, 8};

**int** product = *multiply*(numbers);

System.***out***.println("Product of numbers is: " + product);

}

**public** **void** sayHello() {

System.***out***.println("Hello!");

}

**public** **static** **void** sayBye() {

System.***out***.println("Bye!");

}

**public** **static** **int** multiply(**int**... nums) {

**int** product = 1;

**for** (**int** num : nums) {

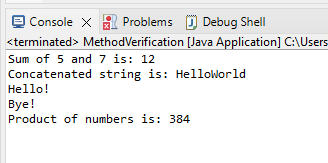
product \*= num;

}

**return** product;

}

}



4. Writing a program in Java to verify the implementations of constructor types

**package** demosample;

**public** **class** ConstructorDemo {

// Instance variables

**private** String name;

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

// Default constructor

**public** ConstructorDemo() {

name = "Unknown";

age = 0;

}

// Parameterized constructor

**public** ConstructorDemo(String name, **int** age) {

**this**.name = name;

**this**.age = age;

}

// Copy constructor

**public** ConstructorDemo(ConstructorDemo obj) {

**this**.name = obj.name;

**this**.age = obj.age;

}

// Getter methods

**public** String getName() {

**return** name;

}

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

**return** age;

}

// Main method

**public** **static** **void** main(String[] args) {

// Create objects using different types of constructors

ConstructorDemo obj1 = **new** ConstructorDemo();

ConstructorDemo obj2 = **new** ConstructorDemo("John", 30);

ConstructorDemo obj3 = **new** ConstructorDemo(obj2);

// Display object details

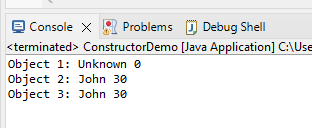
System.***out***.println("Object 1: " + obj1.getName() + " " + obj1.getAge());

System.***out***.println("Object 2: " + obj2.getName() + " " + obj2.getAge());

System.***out***.println("Object 3: " + obj3.getName() + " " + obj3.getAge());

}

}



5. Writing a program in Java to verify implementations of collections.

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.HashSet;

**public** **class** CollectionsDemo {

**public** **static** **void** main(String[] args) {

// ArrayList

ArrayList<String> list = **new** ArrayList<String>();

list.add("Apple");

list.add("Banana");

list.add("Cherry");

System.***out***.println("ArrayList: " + list);

// HashSet

HashSet<String> set = **new** HashSet<String>();

set.add("Dog");

set.add("Cat");

set.add("Horse");

System.***out***.println("HashSet: " + set);

// HashMap

HashMap<String, String> map = **new** HashMap<String, String>();

map.put("USA", "Washington DC");

map.put("Canada", "Ottawa");

map.put("India", "New Delhi");

System.***out***.println("HashMap: " + map);

// Iterate over ArrayList

System.***out***.println("\nIterating over ArrayList:");

**for**(String item : list) {

System.***out***.println(item);

}

// Iterate over HashSet

System.***out***.println("\nIterating over HashSet:");

**for**(String item : set) {

System.***out***.println(item);

}

// Iterate over HashMap

System.***out***.println("\nIterating over HashMap:");

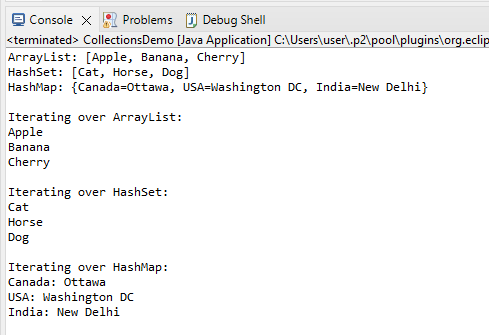
**for**(String key : map.keySet()) {

System.***out***.println(key + ": " + map.get(key));

}

}

}



6 Writing a program in Java to verify implementations of maps

**import** java.util.HashMap;

**import** java.util.Map;

**public** **class** MapsDemo {

**public** **static** **void** main(String[] args) {

// Create a HashMap

Map<String, Integer> map = **new** HashMap<String, Integer>();

// Add elements to the map

map.put("John", 30);

map.put("Jane", 25);

map.put("Alice", 35);

// Display the contents of the map

System.***out***.println("HashMap: " + map);

// Get the value of a key

**int** age = map.get("Jane");

System.***out***.println("Jane's age: " + age);

// Check if a key exists

**boolean** exists = map.containsKey("Bob");

System.***out***.println("Bob exists in the map: " + exists);

// Remove a key-value pair

map.remove("Alice");

System.***out***.println("HashMap after removing Alice: " + map);

// Iterate over the keys

System.***out***.println("\nIterating over keys:");

**for**(String key : map.keySet()) {

System.***out***.println(key);

}

// Iterate over the values

System.***out***.println("\nIterating over values:");

**for**(**int** value : map.values()) {

System.***out***.println(value);

}

// Iterate over the key-value pairs

System.***out***.println("\nIterating over key-value pairs:");

**for**(Map.Entry<String, Integer> entry : map.entrySet()) {

String key = entry.getKey();

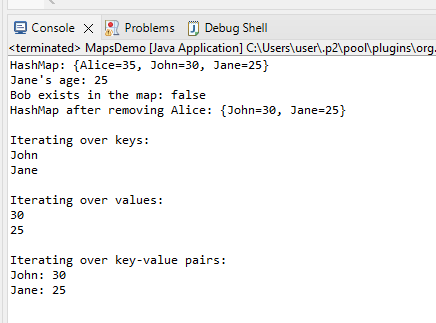
**int** value = entry.getValue();

System.***out***.println(key + ": " + value);

}

}

}



7. Writing a program in Java to verify the implementation of inner classes

**public** **class** InnerClassDemo {

**private** **int** outerVar = 10;

// Inner class

**public** **class** InnerClass {

**public** **void** display() {

System.***out***.println("Outer variable: " + outerVar);

}

}

**public** **static** **void** main(String[] args) {

// Create an instance of the outer class

InnerClassDemo outerObj = **new** InnerClassDemo();

// Create an instance of the inner class

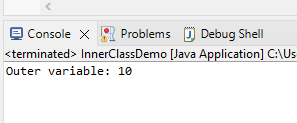
InnerClassDemo.InnerClass innerObj = outerObj.**new** InnerClass();

// Call the display method of the inner class

innerObj.display();

}

}



8. Write a program to create strings and display the conversion of string to StringBuffer and StringBuilder.

public class StringConversionDemo {

public static void main(String[] args) {

// Create a string

String str = "Hello, world!";

// Convert the string to StringBuffer

StringBuffer sb = new StringBuffer(str);

// Convert the string to StringBuilder

StringBuilder sbd = new StringBuilder(str);

// Display the original string

System.out.println("Original string: " + str);

// Display the StringBuffer

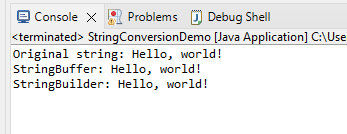
System.out.println("StringBuffer: " + sb);

// Display the StringBuilder

System.out.println("StringBuilder: " + sbd);

}

}



9.Writing a program in Java to verify implementation of arrays

**public** **class** ArrayDemo {

**public** **static** **void** main(String[] args) {

// Declare and initialize an array of integers

**int**[] nums = {1, 2, 3, 4, 5};

// Display the contents of the array

System.***out***.print("Contents of the array: ");

**for** (**int** i = 0; i < nums.length; i++) {

System.***out***.print(nums[i] + " ");

}

System.***out***.println();

// Accessing elements of the array

**int** first = nums[0];

**int** last = nums[nums.length - 1];

System.***out***.println("First element of the array: " + first);

System.***out***.println("Last element of the array: " + last);

// Modifying elements of the array

nums[0] = 10;

nums[2] = 30;

// Display the modified contents of the array

System.***out***.print("Modified contents of the array: ");

**for** (**int** i = 0; i < nums.length; i++) {

System.***out***.print(nums[i] + " ");

}

System.***out***.println();

// Declaring and initializing a 2D array

**int**[][] matrix = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};

// Accessing elements of the 2D array

**int** middle = matrix[1][1];

**int** bottomRight = matrix[2][2];

System.***out***.println("Middle element of the 2D array: " + middle);

System.***out***.println("Bottom right element of the 2D array: " + bottomRight);

// Modifying elements of the 2D array

matrix[0][0] = 10;

matrix[2][1] = 80;

// Display the modified contents of the 2D array

System.***out***.println("Modified contents of the 2D array:");

**for** (**int** i = 0; i < matrix.length; i++) {

**for** (**int** j = 0; j < matrix[i].length; j++) {

System.***out***.print(matrix[i][j] + " ");

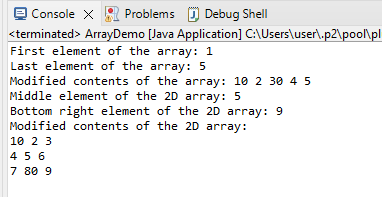
}

System.***out***.println();

}

}

}



10. Writing a program in Java to verify implementations of regular expressions

**import** java.util.regex.\*;

**public** **class** RegexDemo {

**public** **static** **void** main(String[] args) {

// Define a regular expression to match email addresses

String emailRegex = "^[\\w-\\.]+@([\\w-]+\\.)+[\\w-]{2,4}$";

// Define some test email addresses

String[] testEmails = {"johndoe@example.com", "jane.doe@example.co.uk", "invalid-email-address"};

// Create a Pattern object from the regular expression

Pattern pattern = Pattern.*compile*(emailRegex);

// Loop through the test email addresses and match against the pattern

**for** (String email : testEmails) {

Matcher matcher = pattern.matcher(email);

**if** (matcher.matches()) {

System.***out***.println(email + " is a valid email address.");

} **else** {

System.***out***.println(email + " is an invalid email address.");

}

}

}

}

