**Practical Exercise 1 – Basic Java and Object-oriented Programming**

**Overall Objective**

By the end of this practical, you should:

* understand and be able to use the integrated development environment (IDE) for Java
* be able to create, edit, and run your Java programs

**Background**

You will need to know:

1. data types, values and variables
2. input/output
3. selection and iteration
4. methods
5. arrays
6. strings
7. objects and classes
8. inheritance and polymorphism
9. exception handlings
10. abstract classes and interfaces

**Part 1: Discussion**

**Question 1**

Type all of the following code, save the program as specified. Compile the file. Why did it fail?

// saved as Test1.java

public class Testing1 {

public static void main(String args) {

System.out.println("What's wrong with this program?");

}

}

**Answer**: Should be String[] args not String args.

// saved as Test2.java

public class Test2 {

public void main(String[] args) {

System.out.println("What's wrong with this program?")

}

}

**Answer**:

* Should be public static void, not public void.
* Missing semicolon

**Question 2**

What problem arises in compiling the following program?

class A {

public A(int x) {}

}

class B extends A {

public B() {}

}

Answer: The constructor of B did not call the constructor of A. To call the constructor of super(aka parent) class, we need to use super();.

**Question 3**

Show the output of the following programs:

**Program 1**

public class Test {

public static void main(String args[]) {

A a = new A(3);

}

}

class A extends B {

public A(int n) {

System.out.println("A’s constructor is invoked.");

}

}

class B {

public B() {

System.out.println("B’s constructor is invoked.");

}

}

Output:

B’s constructor is invoked.

A’s constructor is invoked.

Reference: [Why is constructor of super class invoked first?](https://stackoverflow.com/questions/7173019/why-is-constructor-of-super-class-invoked-when-we-declare-the-object-of-sub-clas)

**Program 2**

class Base {

public void Base() {

System.out.println("Base");

}

}

public class In extends Base {

public static void main(String argv[]) {

In i = new In();

}

}

No output.

**Program 3**

class Base {

void sub() {

System.out.println("Base");

}

void sub(int p) {

System.out.println(20);

}

}

class Severn extends Base {

public static void main(String argv[]) {

Severn s = new Severn();

s.sub(15);

}

void sub(){

System.out.println("Severn");

}

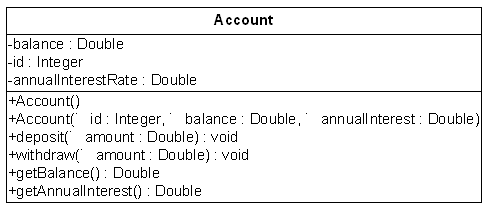
}

Output

20

**Question 4**

Write a class named Account to model accounts. The UML diagram for the class is shown below.

[](https://user-images.githubusercontent.com/23183656/40962702-d60b6466-68d8-11e8-8aa2-475be10aeaae.png)

Write a client program to test the Account class. In the client program, create an Account instance with an account id of 1122, a balance of 20000, and an annual interest rate of 4.5%. Use the withdraw method to withdraw $2500, use the deposit method to deposit $3000, and print the balance and the monthly interest.

public class Account {

private double balance;

private int id;

private double annualInterestRate;

public Account() {

this.balance = 0;

this.id = 0;

this.annualInterestRate = 0;

}

public Account(int id, double balance, double annualInterest) {

this.id = id;

this.balance = balance;

this.annualInterest = annualInterest;

}

public void deposit(double amount) throws Exception {

if(amount <= 0) {

throw new Exception("Cannot deposit negative amount");

}

this.balance += amount;

}

public void withdraw(double amount) throws Exception {

if(amount > this.balance) {

throw new Exception("Insufficient balance");

}

this.balance -= amount;

}

public double getBalance() {

return this.balance;

}

public getAnnualInterest() {

return this.annualInterestRate

}

}

public class Test {

public static void main(String[] args) {

Account myAccount = new Account(1122, 20000, 0.045);

myAccount.withdraw(2500);

myAccount.deposit(3000);

System.out.writeln(myAccount.getBalance());

System.out.writeln(myAccount.getBalance() \* myAccount.getAnnualInterest() / 12);

}

}

**Question 5**

Declare Two classes according to the following instructions:

**(a) Create a base class named Rectangle**

* Declare 2 private object attributes: length and width, that contain in class Rectangle.
* Declare a public constructor that takes 2 parameters (init\_length, init\_width) that populates the length and width attributes.
* Declare a public method area() that returns that surface area of the rectangle.

public class Rectangle {

private double length;

private double width;

public Rectangle(double init\_length, double init\_width) {

this.length = init\_length;

this.width = init\_width;

}

public double area() {

return this.length \* this.width;

}

}

**(b) From the base class Rectangle, create a subclass named Box.**

* Declare an additional object attribute named: depth
* Declare a constructor for the derived Box class that takes 3 parameters (init\_length, init\_width, init\_depth) that populates the length, width and depth attributes.
* Declare an override method named area() that returns that surface area of the box.
* Declare a method named volume() that returns that volume of the box.

(Formula for computing: area of a rectangle, area = length × width area of a box , area = 6 × length × width volume of a box, volume = length × width × height)

public class Box extends Rectangle {

private double depth;

public Box(double init\_length, double init\_width, double init\_depth) {

super(init\_length, init\_width);

this.depth = init\_depth;

}

@Override

public double area() {

return super.area() \* 6;

}

public double volume() {

return super.area() \* this.depth;

}

}

**(c) Write a client program to test both Rectangle and Box classes.**

* Create an instance that stores a 10 × 12 rectangle. Print the area of the rectangle.
* Create an instance that stores a 10 × 12 × 8 box. Print the area and volume of the box.

public class Test {

public static void main(String[] args) {

Rectangle myRectangle = new Rectangle(10, 12);

System.out.println(myRectangle.area());

Box myBox = new Box(10, 12, 8);

System.out.println(myBox.area());

System.out.println(myBox.volume());

}

}

**Question 6**

Given that :

public abstract class Lecturer {

private String name;

private int id;

public Lecturer (String name, int id) {

this.name = name;

this.id = id;

}

public void setName(String name) {

this.name = name;

}

public String getName() {

return name;

}

public void setId(int id) {

this.id = id;

}

public int getId() {

return id;

}

public String toString() {

return getName( ) + " " + getId( );

}

public abstract double salary();

}

From the base class Lecturer, create a subclass named PartTimeLecturer with the following requirements.

* Declare an additional double typed attribute named: teachingHour.
* Declare 3 constructors for the derived PartTimeLecturer class
  + The first constructor should be a default constructor.
  + The second constructor should take one parameter that populates the id attribute.
  + The third constructor should take three parameters that populate the name, id and teachingHour attributes.
* Declare a mutator method named setTeachingHour( ) that sets a lecturer’s teaching hours.
* Declare an accesor method named getTeachingHour( ) that returns an lecturer’s teaching hours.
* Declare an override method named toString( ) that returns a lecturer’s name, id and teachingHour.
* Implement the abstract method named salary( ) declared in class Lecturer that returns the part time lecturer’s salary corresponding to the teaching hours. The salary can be determined using the formula: salary = teachingHour × 100.00;

public class PartTimeLecturer extends Lecturer {

private double teachingHour;

public PartTimeLecturer() {

super(null, 0);

this.teachingHour = 0;

}

public PartTimeLecturer(int id) {

super(null, id);

this.teachingHour = 0;

}

public PartTimeLecturer(String name, int id, double teachingHour) {

super(name, id);

this.teachingHour = teachingHour;

}

public void setTeachingHour(double newTeachingHour) {

this.teachingHour = newTeachingHour;

}

public double getTeachingHour() {

return this.teachingHour;

}

@Override

public String toString() {

return "Name: " + super.getName() + "\n" +

"Id: " + super.getId() + "\n" +

"Teaching hour: " + this.teachingHour + "\n";

}

public double salary() {

return this.teachingHour \* 100.00;

}

}

Write a client program to test both Lecturer and PartTimeLecturer classes.

public class Test {

public static void main(String[] args){

Lecturer john = new PartTimeLecturer("John", 999, 10);

System.out.println(john);

}

}

**Question 7**

Given

public class Vehicle {}

public interface VehicleMethods {

public double getWeight();

public int getSpeed();

}

(i) Write an abstract class LandVehicle that inherits from the class Vehicle and implements the VehicleMethods interface.

public abstract class LandVehicle extends Vehicle implements VehicleMethods {

// Empty body

}

(ii) Write a subclass Car that inherits from LandVehicle and implements the two methods declared in the interface VehicleMethods. Assume there are no other attributes and methods in both class declarations. The constructors can also be omitted.

public class Car extends LandVehicle {

private double weight;

private int speed;

public Car(double weight, int speed) {

this.weight = weight;

this.speed = speed;

}

public double getWeight() {

return this.weight;

}

public int getSpeed() {

return this.speed;

}

}