**Examination Answer Book**

**UNIVERSITY EXAMS**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| REGISTRATION NUMBER | | | | | | | | | VU-DIT-2301-0474-EVE | | | | | | |
| Title of The Program (eg BBA, BSC, BPH, BSWA) | | | | | | | | | | | | | DIT | | |
| Diploma in Information Technology | | | | | | | | | | | | | | | |
| Department | | | | Other Depts in Faculty of Science and Technology | | | | | | | | | | | |
| Faculty | Faculty of Science and Technology | | | | | | | | | | | | | | |
| Year Of study (YrI , YrII, YrIII, or YrIV) | | | | | | | | | | | 2 | | | | |
| Module Code and Name | | | | | | | 1303 FST | | | | | | | | |
| Object Oriented Programming | | | | | | | | | | | | | | | |
| Semester | | | 1 | | | | | | | | | | | | |
| (Copy from the heading to the Examination Paper) | | | | | | | | | | | | | | | |
| Retake: | | Yes | | |  | | | No | |  | | (Tick whichever is applicable) | | | |
| Date of examination | | | | | | Fri Jun 07 2024 09:00:00 GMT+0300 (East Africa Time) | | | | | | | | | |
|  | | | | | |  | | | | | | | | | |
| **DIRECTIONS TO CANDIDATES (Turn to page ii for more instructions).** | | | | | | | | | | | | | **FOR USE BY EXAMINERS ONLY** | | |
| **Question Number** | **Internal Examiner** | **External Examiner** |
| 1. Leave margin blank. 2. Begin each answer on a fresh page. 3. Write the number of each question and theCandidate's Number at the top of each page. 4. Write the numbers of the questionswhich you have attempted, with subsections where necessary, in the spacesprovided below | | | | | | | | | | | | |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **TOTAL** |  |  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **NUMBER OF QUESTIONS** you have answered in the order in which you have written them | | | | | | | | |
| 3 | 5 | 6 | 2 |  |  |  |  |  |

**How and where should I submit my examination script?**

Every student will be required to attend their examination via [VClass Students Portal](https://vclass.ac/) E.g. you go to [www.vclass.ac](http://www.vclass.ac) and login, to your account, then on the left sidebar menu **click on Examinations**.

Under examinations you will see the following: -

1. Instructions for that particular examination with time required to finish your examination as per instructions,
2. A student will be required to download the question paper and the answer sheet provided by the university within the same module examination, or a student can be required to attempt structured questions within the system depending on how the examination was set.
3. Submission of answered questions is done,
4. Student is required to click to **consent** to show that the answered exam belongs to them.
5. **Note** that if an examination is for download, a student will be required to download the question paper and answer sheet, write their examination within the given stipulated time.
6. Required to scan and upload back the answered booklet through the same portal as per format available.
7. Examinations uploaded will directly be received by the Registry department.
8. Students here are required to use [VClass e-Learning system](https://vclass.ac)for all examinations and for any failure they can contact the Registry department for guidance.
9. No late submission will be accepted.

**Avoid any examination malpractice because this will attract severe penalties such as invalidating the exams answered script whose consequences will attract retakes.**

**MURUNGI ISMAIL**

Question 3

1. the purpose of access modifiers in OOP languages

Access modifiers in OOP languages are used to define the accessibility of classes, methods, and variables. They help in encapsulating data and ensuring that the implementation details are hidden from other parts of the program. They include the following;

public: The member is accessible from any other class.

private: The member is accessible only within its own class.

protected: The member is accessible within its own package and by subclasses.

Default: The member is accessible only within its own package.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ACCESS MODIFIER | PUBLIC | PROTECTED | DEFAULT | PRIVATE |
| DEFINING CLASS | YES | YES | YES | YES |
| CLASS IN SAME PACKAGE | YES | YES | YES | NO |
| SUBCLASS IN DIFFERENT PACKAGE | YES | YES | NO | NO |
| NON-SUBCLASS IN DIFFERENT PACKAGE | YES | NO | NO | NO |

2. IMPLEMENTATION OF PREFERENCE

import java.util.HashMap;

import java.util.Map;

// Custom exception for when preference key is not found

class PreferenceNotFoundException extends Exception {

public PreferenceNotFoundException(String message) {

super(message);

}

}

public class Preference {

// Map to store preferences

private Map<String, String> preferences;

// Constructor initializes the preferences map

public Preference() {

preferences = new HashMap<>();

}

// Set or update a preference

public void setPreference(String key, String value) {

preferences.put(key, value);

}

// Retrieve a preference

public String getPreference(String key) throws PreferenceNotFoundException {

if (!preferences.containsKey(key)) {

throw new PreferenceNotFoundException("Preference with key '" + key + "' not found.");

}

return preferences.get(key);

}

}

The Preference class provides a simple implementation to store game preferences using a key-value mapping.

Preferences are stored in a Map<String, String> where each preference is associated with a unique string key.

The setPreference() method allows preferences to be set or updated by providing a key and its corresponding value.

The getPreference() method retrieves a preference value given its key. If the key is not found in the preferences map, it throws a PreferenceNotFoundException.



how the Singleton design pattern is applied to ensure only one Preference object exists in the running game

public class Preference {

// Singleton instance

private static Preference instance;

// Map to store preferences

private Map<String, String> preferences;

// Private constructor to prevent instantiation from outside

private Preference() {

preferences = new HashMap<>();

}

// Lazily instantiate the object and ensure only one instance exists

public static synchronized Preference getInstance() {

if (instance == null) {

instance = new Preference();

}

return instance;

}

// Set or update a preference

public void setPreference(String key, String value) {

preferences.put(key, value);

}

// Retrieve a preference

public String getPreference(String key) throws PreferenceNotFoundException {

if (!preferences.containsKey(key)) {

throw new PreferenceNotFoundException("Preference with key '" + key + "' not found.");

}

return preferences.get(key);

}

}

The Singleton pattern ensures that only one instance of the Preference class exists throughout the program's execution.

The class provides a private constructor to prevent external instantiation.

The getInstance() method lazily instantiates the Preference object if it hasn't been created yet. It ensures thread safety by synchronizing the method.

Access modifiers are appropriately applied. The constructor is private to prevent external instantiation, and the methods are public to allow access from other parts of the program.

Implementing the Cloneable interface allows objects of a class to be cloned. However, in the Singleton pattern, the goal is to have only one instance of the class throughout the program's execution. Allowing cloning would violate this principle, as cloning would create a new instance of the class, thereby defeating the purpose of the Singleton pattern. Therefore, implementing the Cloneable interface is not appropriate for a class designed to be a Singleton.

**QUESTION FIVE**



import java.util.Scanner;

public class BukotoVillageBank {

public static void main(String[] args)

{

Scanner scanner = new Scanner(System.in);

System.out.print("Enter number of members: ");

int numMembers = scanner.nextInt();

double[] principalAmounts = new double[numMembers];

double[] interestRates = new double[numMembers];

int[] timeFrames = new int[numMembers];

// Using for loop to input data

for (int i = 0; i < numMembers; i++) {

System.out.print("Enter principal amount for member " + (i + 1) + ": ");

principalAmounts[i] = scanner.nextDouble();

System.out.print("Enter interest rate for member " + (i + 1) + ": ");

interestRates[i] = scanner.nextDouble();

System.out.print("Enter time frame for member " + (i + 1) + ": ");

timeFrames[i] = scanner.nextInt();

}

// Using while loop to calculate and display investment values

int i = 0;

while (i < numMembers) {

double investmentValue = calculateInvestmentValue(principalAmounts[i], interestRates[i], timeFrames[i]);

System.out.println("Investment value for member " + (i + 1) + ": " + investmentValue);

i++;

}

}

private static double calculateInvestmentValue(double principalAmount, double interestRate, int timeFrame) {

return principalAmount \* Math.pow((1 + interestRate), timeFrame);

}

}



import java.util.Scanner;

public class AlexaAndBalexisMerchants {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter first number: ");

double num1 = scanner.nextDouble();

System.out.print("Enter second number: ");

double num2 = scanner.nextDouble();

System.out.print("Enter third number: ");

double num3 = scanner.nextDouble();

double product = num1 \* num2 \* num3;

double sum = num1 + num2 + num3;

double max = Math.max(Math.max(num1, num2), num3);

System.out.println("Product\tSum\tMaximum");

System.out.println("-----\t----\t------");

System.out.printf("%.0f\t%.0f\t%.0f%n", product, sum, max);

QUESTION SIX

Number 6:

(a)

(i) super() calls the constructor of the superclass (Object) to ensure it's initialized before the Element class.

(ii) this.item = item assigns the passed item parameter to the item field of the current Element object (this).

(iii) @Override indicates that the toString() method is overriding the one from the superclass (Object).

(iv) To make the Element class immutable, i remove the setter methods and make the fields final:

class Element {

final int item;

final Element next;

Element(int item, Element next) {

this.item = item;

this.next = next;

}

@Override

public String toString() {

return item + (next == null ? ''': next);

}

}

b)

Implementation of the immutable FuncList class:

class FuncList {

private final Element head;

FuncList() {

head = null;

}

FuncList(int item, FuncList tail) {

head = new Element(item, tail.head);

}

int head() {

return head.item;

}

FuncList tail() {

return new FuncList(head.next);

}

FuncList cons(int item) {

return new FuncList(item, this);

}

}

(c)

(i) The generic FuncList<T> is no longer immutable because the type parameter T could be a mutable type. To solve this, i ensure that T is an immutable type or i use a wrapper class to make it immutable.

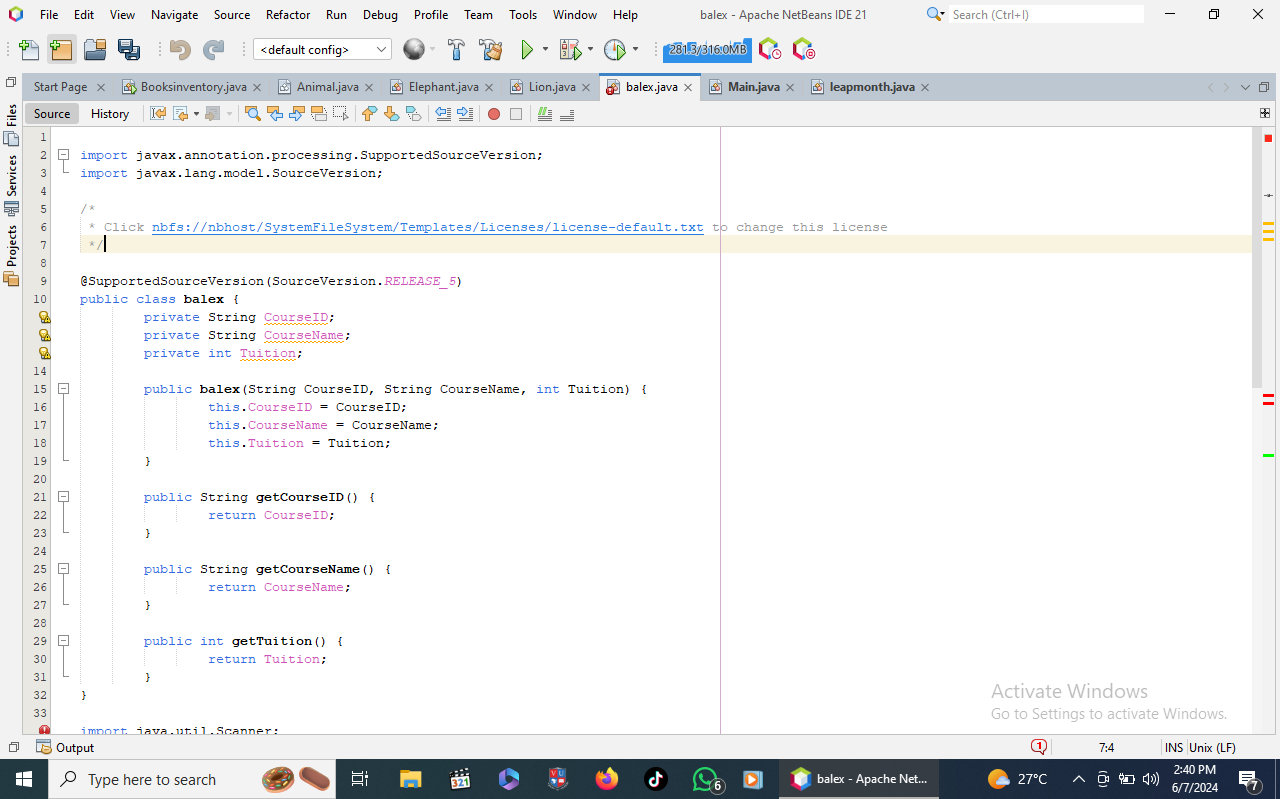
(ii) Java's restriction on covariance is necessary to prevent errors as in the program below:

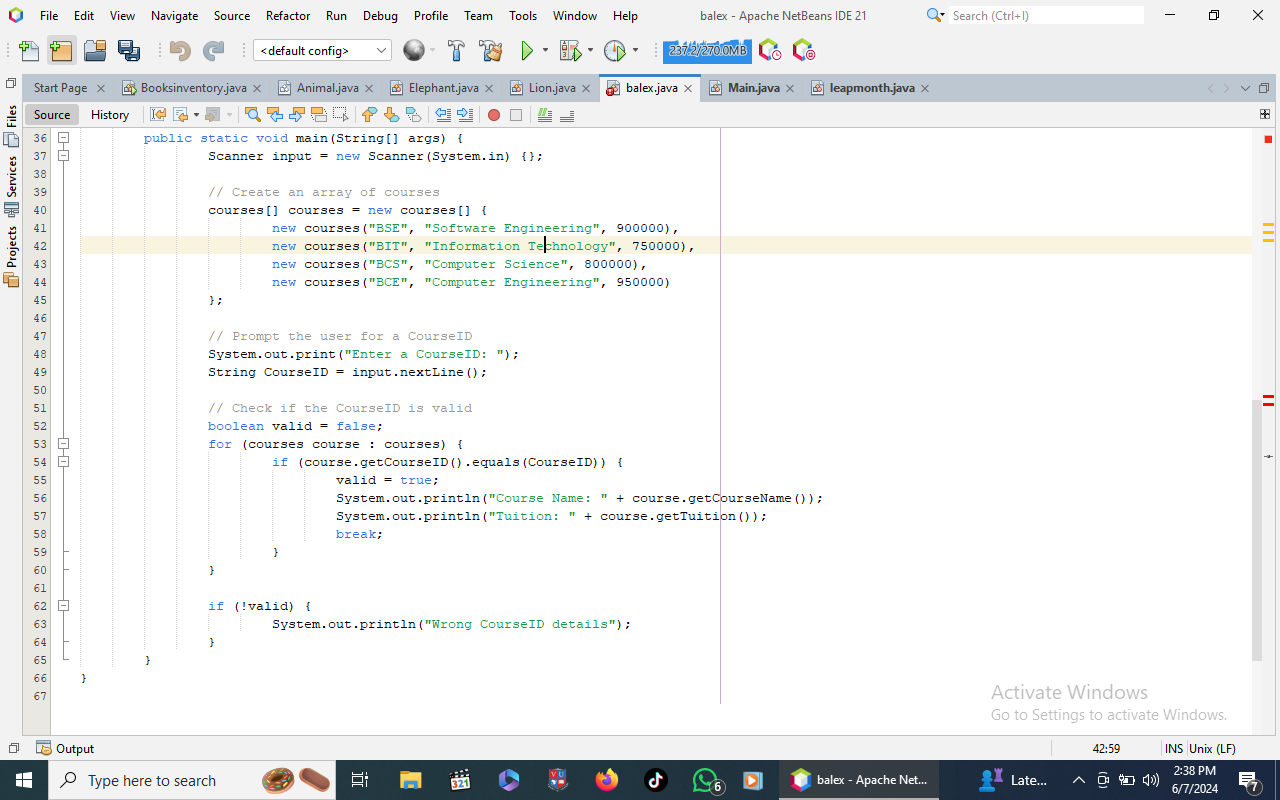
ncList<String> stringList = new FuncList<>("hello");

FuncList<Object> objectList = stringList; // covariance would allow this

objectList.cons(new Integer(42)); // would add an Integer to a String list

QUESTION 2





https://github.com/murungi-ismail/Question-2-balex