Faculty of Computing

SE1020 – Object Oriented Programming

Year 1 Semester 2 (2025)

Tutorial 01

1. Define Object-Oriented Programming (OOP). How does it differ from Procedural Programming?

2. Identify Objects in an Online Shopping System.

   Consider an Online Shopping System. Identify at least three classes that would exist in such a system. For each class, define at least three attributes and two methods.

3. Write a Java program to calculate a student's final grade as follows.

   (a) Create a class called **Student** with the following private attributes
   - studentName
   - assignmentMarks (out of 100)
   - examMarks (out of 100)
   - finalGrade (determined based on total marks)

   (b) Use getter and setter methods to:
   - Set and retrieve the student's name.
   - Set and retrieve the assignment marks.
   - Set and retrieve the exam marks.

   (c) Calculate the final grade using the formula.

   $$\textbf{Total Marks} = (\textbf{assignmentMarks} * \textbf{0.4}) + (\textbf{examMarks} * \textbf{0.6})$$

   (d) Assign the final grade based on total marks.
   - A $\rightarrow$ Total Marks ≥ 80
   - B $\rightarrow$ Total Marks ≥ 65 and < 80
   - C $\rightarrow$ Total Marks ≥ 50 and < 65
   - F $\rightarrow$ Total Marks < 50

   (e) Create a main method to:
   - Accept user input for student details.
   - Use setter methods to assign values.
   - Use getter methods to retrieve and print the final grade.

4. Develop a system to manage bank accounts for a banking application.

   Scenario:

   A bank wants to create a simple system to manage its customers' savings accounts. Each customer has a savings account with details such as account number, account holder's name, and account balance. The bank also needs to ensure that account details like balance are secure and cannot be accessed directly. The bank requires that the system:

   - Protects the account balance by restricting direct access to it.
   - Provides methods (getters and setters) to retrieve the balance and allow deposits and withdrawals, ensuring the balance cannot become negative.

   (a) Create a class **SavingsAccount** with necessary attributes.

   (b) Implement getters and setters for **accountNumber** and **accountHolderName**.

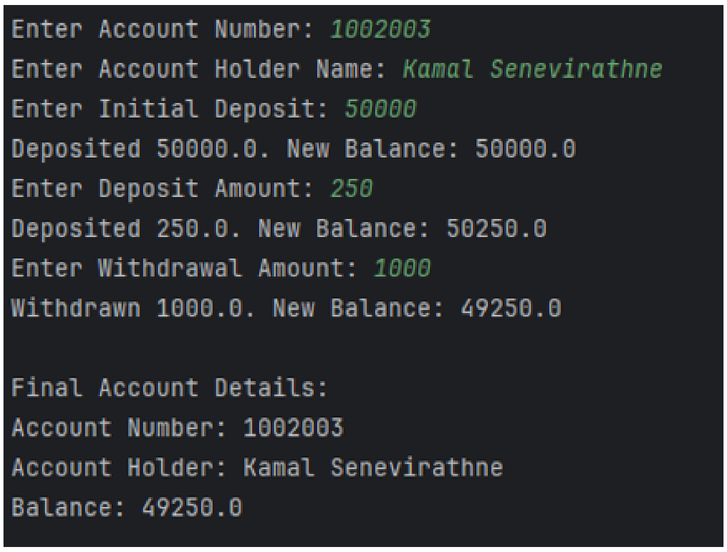
   (c) Create a getter for **balance** (No setter for balance).

   (d) Implement a method called **deposit(double amount)** that adds amount to balance (amount should be positive).

   (e) Implement a method called **withdraw(double amount)** that deducts amount from balance only if there are sufficient funds.

   (f) Create a **displayAccountDetails()** to print account details.

   Sample Output:



```
Enter Account Number: 1002003
Enter Account Holder Name: Kamal Senevirathne
Enter Initial Deposit: 50000
Deposited 50000.0. New Balance: 50000.0
Enter Deposit Amount: 250
Deposited 250.0. New Balance: 50250.0
Enter Withdrawal Amount: 1000
Withdrawn 1000.0. New Balance: 49250.0

Final Account Details:
Account Number: 1002003
Account Holder: Kamal Senevirathne
Balance: 49250.0
```

Faculty of Computing

SE1020 – Object Oriented Programming

Year 1 Semester 2 (2025)

Tutorial 02

Question 01)

A university wants to build a system to manage student details. Each student has a **student ID, name**, and **GPA**. The **GPA should be between 0.0 and 4.0**. Ensure that these details are properly secured and only accessible through appropriate methods.

a) Create a class named **Student** with studentId (int), name (String), and gpa (double) as attributes.

b) Implement **getters and setters** for each attribute with the following **validations**:

- **GPA** must be **between 0.0 and 4.0**. If an invalid GPA is entered, display an error message.

c) Create a method named **displayStudentDetails()** to display the student's details.

d) Create another class called **StudentApp** with the main method and;

- Create a **Student object**.
- Accept **user input** for the student's ID, name, and GPA using **setters**.
- **Display the student's details** using **displayStudentDetails()**.

Question 02)

You are building a geometry calculator to calculate the area of a circle. Depending on the data available, the area can be calculated in **two ways** as:

- Using the radius of the circle.
- Using the diameter of the circle.

a) Create a class named **Circle** with radius (double) as an attribute.

b) Implement **two overloaded methods named calculateArea()**:

- **Method 1**: Accepts **no parameters** – returns the area using the **radius attribute**.
- **Method 2**: Accepts **the diameter as a parameter** – calculates and returns the area.

c) Create **getters** and **setters** for **radius**.

d) Implement another class called **CircleApp** with the **main method** and;

- Create a **Circle object** .
- Accept the **radius from the user** .
- Display **the area using the radius** .
- Accept the **diameter from the user** and **display the area using the diameter** .

[Hint: Area = (22 / 7.0) * radius * radius]

Question 03)

A car showroom wants a system to manage car details. Each car has a **registration number, model name**, and **price**.

a) Create a class named **Car** with regNumber (String), modelName (String), and price (double) as attributes

b) Implement the following a **default constructor** and a **parameterized constructor**.

c) Create g**etters and setters** for all attributes.

d) Implement **displayCarDetails()** method to display car information.

e) Implement another class called **CarApp** with the **main method** and:

- Create a **Car object using the default constructor** and **display details**.
- Create **another Car object using the parameterized constructor** and **display details**.

Question 04)

An electricity company wants a system to calculate the monthly **electricity bill** of customers. Each customer has a **customer ID, name**, and **units consumed**. The bill is calculated based on the **units consumed**:

a) Create a class named **Customer** with customerId (int), name (String), and unitsConsumed (int) as attributes.

b) Implement a **Parameterized constructor**.

c) **Getters and setters** for each attribute.

d) Implement a method called **calculateBill()** which returns the bill amount based on the units consumed:

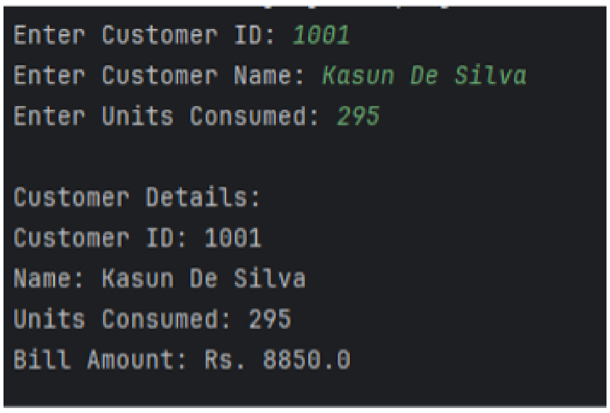
- Up to 100 units → Rs. 20 per unit
- 101 to 300 units → Rs. 30 per unit
- Above 300 units → Rs. 40 per unit

e) Implement a method called **displayCustomerDetails()** to display the **Customer ID, Name, Units Consumed, and Bill Amount**.

f) Create a class called **CustomerApp with the main method** and;

  a. **Accept user input** for the **customer's details**.
  b. **Display the customer's details**, including the **calculated bill amount**.

Sample Output:



```
Enter Customer ID: 1001
Enter Customer Name: Kasun De Silva
Enter Units Consumed: 295

Customer Details:
Customer ID: 1001
Name: Kasun De Silva
Units Consumed: 295
Bill Amount: Rs. 8850.0
```

Faculty of Computing

SE1020 – Object Oriented Programming

Year 1 Semester 2 (2025)

Tutorial 03

Question 01 – Noun Verb Analysis

"ABC" Television Corporation is a leading television channel in Sri Lanka. The management has decided to automate the activities of their advertising department. Advertising department mainly focus on advertisement scheduling. You have been hired as a system analyst to identify the requirements and to make an initial sketch of the proposed system.

The staff of the advertising department consists with a manager, an assistant manager, a secretary and several designers. Before taking advertisements, ABC Television Corporation signs a contract with the client.

When a new advertisement is received from a client or a client company, the secretary will log in to the system and check the terms of the client contract by client number. Then the details of the advertisement (This may include the length of the advertisement, time belt, payment, client name, client email etc) are entered in to the system. Since a new advertisement is available, secretary has to re-generate the advertisement schedule for the rest of the day. This schedule can be then printed and given to the telecasting unit.

If the client contract is changed due to any circumstance, the relevant advertisement schedule is updating accordingly. this is again done by the secretary.

An assistant manager can log in to the system and can view the advertisements added on a given date. For any reason, if the initially assigned time belt for a particular advertisement should be changed, that can only be done by an assistant manager or a manager.

At the end of each week, the assistant manager can generate the weekly scheduling status report. It visualize the status of the advertisement in the previous week. Frequently reports can be generate per each advertisement, to view the telecasting frequency of a particular advertisement. The client report can be generate at any time to get the details of clients; profit gained by each clients and the client payments. All the reports can be generated either by assistant manager or manager.

Clients can pay for advertisements directly through credit cards. The system is linked with existing accounting module to handle the payment details. An email will be send to the client for confirmation of payment. This will again handle by the accounting module.

Do the Noun Verb analysis to identify the classes in the description.

a) Identify all the Nouns which can be potential classes

b) Identify the set of potential classes, show clearly how the nouns were rejected using the five rules.

c) Draw the CRC cards for any 5 classes you have identified in part (b)

Question 02 – Inheritance

A Transport Company to develop a **Vehicle Management System**. The system manages different types of vehicles, such as cars and trucks, and calculates their rental costs.

The company applies different rental rates based on vehicle type as below:

Vehicle Type	**Rate per Day**
Car	\$50.00
Truck	\$80.00

a) Create a class called **Vehicle** with the vehicleID (String), vehicleName (String), and rentalDays (int) as attributes.

b) Implement a default constructor and a parameterized constructor.

c) Implement a method called *displayDetails()* to show the vehicle information.

d) Create a subclass called **Car** that inherits from Vehicle.

   i. Implement a method *calculateRentalCost()* to calculate the rental cost using \$50.00 per day.

   ii. **Override** the *displayDetails()* method to show the rental cost along with car details.

e) Create another subclass called **Truck** that also inherits from Vehicle.

   i. Implement a method *calculateRentalCost()* to calculate the rental cost using \$80.00 per day.

   ii. Override the *displayDetails()* method to show the rental cost along with vehicle details.

f) In the main() method, create objects of both Car and Truck, , and display their details using the overridden methods.

Question 03 – Inheritance and Polymorphism

Implement an **Online Course Enrollment** System that manages different types of courses and calculates the total course fees for students. The platform offers both Regular Courses and Premium Courses. Premium courses include additional certification services, while regular courses do not.

a) Create a class called **Course** with courseID (String), courseName (String), and durationWeeks (int) as attributes.

b) Create a default constructor and a parameterized constructor to assign all details of a Course.

c) Implement a method called *displayCourseDetails()* to print course information.

d) Implement method overloading as below.

   i. Implement *calculateTotalFee()* with no parameters to return 10000.00as the basic course fee.

   ii. Overload *cacalculateTotalFee* method with one double parameter as *discount-Percentage* to calculate the total fee using the discount value.
   [Hint: call the calculateTotalFee() method get the basic course fee]

e) Create a subclass called **RegularCourse** that inherits from Course class and do followings.

   i. Add an additional attribute hasLiveSessions (boolean) to indicate whether live sessions are included.

   ii. Implement a default constructor and a parameterized constructor constructor to assign all details of a Regular Course (including coursed, courseName and durationWeeks).

   iii. Override the calculateTotalFee() method to compute the total fee as;

$$\text{Total Fee} = 100 \times \text{durationWeeks}$$

   iv. Override the *displayCourseDetails()* method to display all the details including the total fee and live session status.

f) Create another subclass called **PremiumCourse** that inherits from Course and do followings.

   i. Add an additional attribute called includeCertification (boolean).

   ii. Implement a default constructor and a parameterized constructor constructor to assign all details of a Premium Course (including coursed, courseName and durationWeeks).

   iii. Override the calculateTotalFee() method to compute the total fee as;

$$\text{Total Fee} = (150 \times \text{durationWeeks}) + (\text{Certification Fee of \$50 if includeCertification is true})$$

iv. Override the *displayCourseDetails()* method to include the total fee and certification status.

g) Implement the main method as follows;

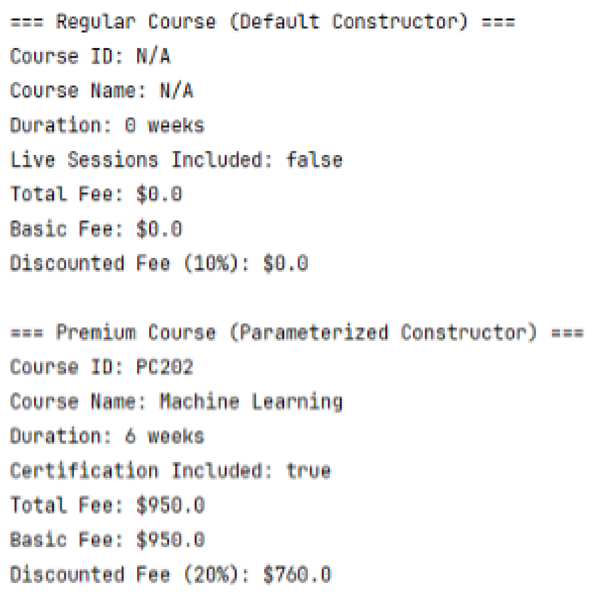
i. Create one object from **RegularCourse** using the default constructor.

ii. Create one **PremiumCourse** object using parameterized constructors.

iii. Call both overloaded versions of calculateTotalFee() for the objects.

iv. Display all course details using the overridden displayCourseDetails() method (refer to the output given below).

Sample Output:



```
=== Regular Course (Default Constructor) ===
Course ID: N/A
Course Name: N/A
Duration: 0 weeks
Live Sessions Included: false
Total Fee: $0.0
Basic Fee: $0.0
Discounted Fee (10%): $0.0

=== Premium Course (Parameterized Constructor) ===
Course ID: PC202
Course Name: Machine Learning
Duration: 6 weeks
Certification Included: true
Total Fee: $950.0
Basic Fee: $950.0
Discounted Fee (20%): $760.0
```

Question 04 – Inheritance and Polymorphism

Develop a Library Management System that tracks books and their authors. Each book is written by an author, and each author has specific details like name and nationality. Each book contains an Author object as **a part of** its details. An author **cannot exist** inside the system without a book. The author's details are meaningful only when associated with a book. If a Book object is deleted, the Author object contained inside it is also will get deleted.

a) Create a class called Author with authorName (String), nationality (String) as attributes and implement the following in the Author class:

i. A default constructor and a parameterized constructor.

ii. A method *displayAuthorDetails()* to print author information.

b) Create a class called **Book** that contains an author (Author), bookID (String), title (String), price (double), stockQuantity (int) as attributes. Implement the following in the Book class:

   i. A default constructor and a parameterized constructor (initialize the Author object inside it).

   ii. A method to print book and author details.

c) Implement below 2 methods supporting method overloading in the Book class:

   i. isAvailable() with no parameters → Returns true if stockQuantity is greater than 0, otherwise false.

   ii. isAvailable(int requestedQuantity) → Returns true if requestedQuantity less than or equal to stockQuantity, otherwise false.

d) Implement the main() method as below.

   i. Create at least one Author object.

   ii. Create at least two Book objects (one using the default constructor, another using the parameterized constructor).

   iii. Call both overloaded versions of *isAvailable()*.

   iv. Display all book details using *displayBookDetails()*.

SLIIT UNI
THE KNOWLEDGE UNIVERSITY

Faculty of Computing

SE1020 – Object Oriented Programming

Year 1 Semester 2 (2025)

Tutorial 04

Question 01

In a Smartphone, the Battery and Processor are essential components that cannot function independently outside the device. The Smartphone fully owns and controls these components, meaning their existence depends on the Smartphone. If the Smartphone is destroyed, the Battery and Processor are also destroyed.

a) Create a class called **Battery** with the attribute capacity (int) as an attribute and implement a method to display battery details.

b) Create another class called **Processor** with model (String) as an attribute and implement a method to display processor details.

c) Implement a class called **Smartphone** that contains a Battery and a Processor.

   i. Initialize Battery and a Processor instances inside the Smartphone constructor.

   ii. Provide a method to display full smartphone details, including battery and processor information

d) Create a class called **SmartApp** with the main method to:

   i. Create an object of Smartphone (with a Battery and Processor).

   ii. Display all details.

Sample Output:

```
Smartphone Brand: Samsung Galaxy
Battery Capacity: 5000mAh
Processor Model: Snapdragon 888
```

Question 02

A Car is a type of Vehicle, meaning it inherits common properties of a Vehicle such as brand and speed. At the same time, a Car is composed of an Engine, which means the Engine is a necessary component that cannot exist independently outside the Car.

a) Create a class called **Vehicle** with brand (String), and speed (int) as attributes. Then implement a method to display vehicle details.

b) Create a **Car** class that extends Vehicle and contains model (String) as an additional attribute.

   i. Implement a parameterized constructor to initialize all the attributes.

   ii. Creae a method to display car details.

c) Implement a class called **Engine** with engineType (String) as an attribute. Then implement a method to display engine details.

d) As the Engine is a part of the Car and Engine cannot exist without a Car, modify the Car class to include an Engine instance. Initialize Engine inside the Car's constructor.

e) Modify the display method to display full car details, including engine information.

f) Create a class called **CarApp** with a main method to:

   i. Create a Car object (which includes an Engine).

   ii. Display all details.

Question 03

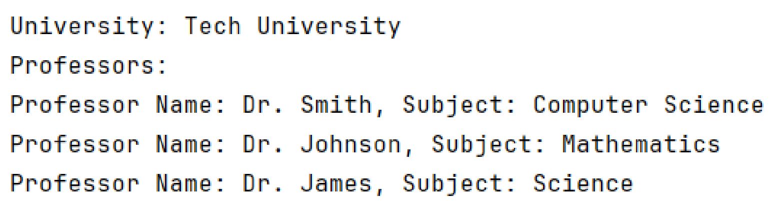
A University has multiple Professors who specialize in different subjects. Each Professor exists independently and is not owned by the University. If the University closes, the Professors can continue to work elsewhere. The University should store multiple Professor objects using a simple array and allow displaying all associated details.

a) Write a Java program to create a class called **Professor** with attributes for name and subject, along with a method to display professor details.

b) Implement a class called **University** that stores multiple Professor objects using a simple array and provides a method to display all university details, including the associated professors.

c) Create a main method in a **UniversityApp** class to create multiple Professor objects separately, assign them to a University, and display all details.

Sample Output:



```
University: Tech University
Professors:
Professor Name: Dr. Smith, Subject: Computer Science
Professor Name: Dr. Johnson, Subject: Mathematics
Professor Name: Dr. James, Subject: Science
```

Question 04

A Transport Management System manages different types of vehicles used in a transportation company. Each vehicle has a brand and speed, but there are specialized vehicles such as Buses and Trucks that have unique features.

Each Vehicle contains an Engine that determines its power, meaning Engine is a part of the Vehicle and cannot exist without it.

Additionally, the transportation company owns multiple Drivers, and each Driver is associated with a Vehicle. However, a Driver can exist independently of a specific vehicle.

Different types of vehicles have unique implementations of the start() method, which need to be overridden in the respective classes

a) Implement a class called **Vehicle** with brand(String) and speed(double) attributes.

   i. Create a method called start(), that prints a generic message.

b) Create two classes called **Bus** and **Truck** as subclasses of Vehicle.

   i. Override the start() method to display unique messages for each type.

c) Implement an **Engine** class with engineType(String) as an attribute.

   i. The Engine should be part of a Vehicle (Composition) and initialized within the Vehicle constructor.

d) Create a class called **Driver** with name(String) and licenseNumber(String) attributes.

   i. A Driver should be associated with a Vehicle using Aggregation

e) Modify the **Bus** and **Truck** classes to include a Driver for each class(Aggregation).

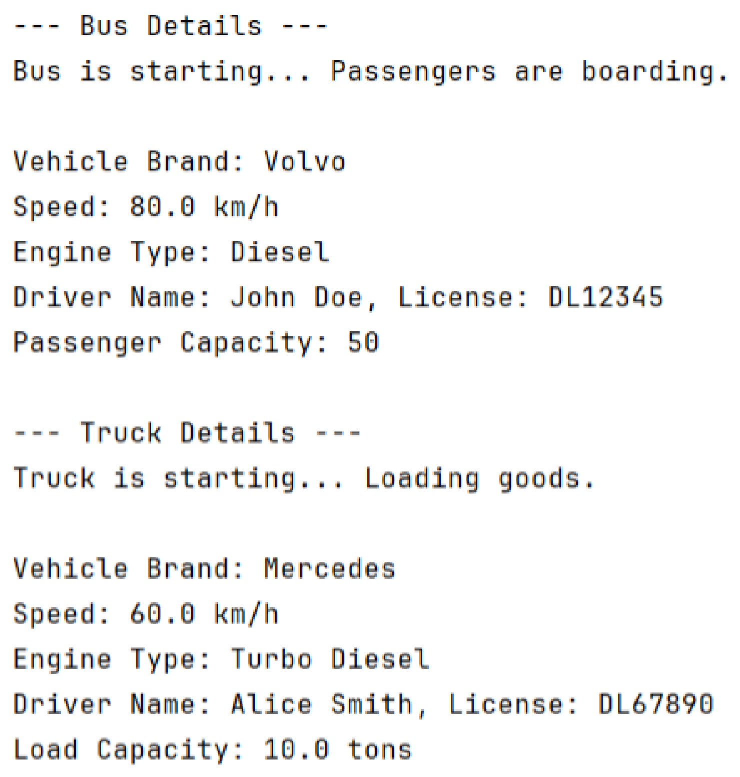
f) Create a **TransportApp** class with a main method to:

   i. Create objects for different Vehicles (Bus, Truck) with their Engines.

   ii. Assign Drivers to those Vehicles.

   iii. Start the vehicles and display full details, including the associated Driver and Engine.

Sample Output:



```
--- Bus Details ---
Bus is starting... Passengers are boarding.

Vehicle Brand: Volvo
Speed: 80.0 km/h
Engine Type: Diesel
Driver Name: John Doe, License: DL12345
Passenger Capacity: 50

--- Truck Details ---
Truck is starting... Loading goods.

Vehicle Brand: Mercedes
Speed: 60.0 km/h
Engine Type: Turbo Diesel
Driver Name: Alice Smith, License: DL67890
Load Capacity: 10.0 tons
```

Faculty of Computing

SE1020 – Object Oriented Programming

Year 1 Semester 2 (2025)

Tutorial 05

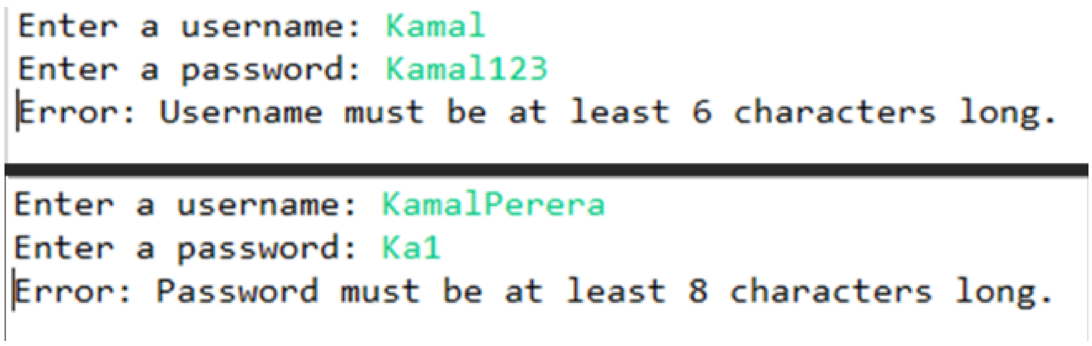
Question 01

Write a program that prompts the user to enter a username and password, then perform the following validations using two different custom exception classes as follows.

- **InvalidUserName**, is an exception class that prints out the error message "The username must be at least 6 characters long" if the number of characters in the user input for the username is less than 6.
- **InvalidPasswordLength**, is an exception class that prints out the error message "The password must be at least 8 characters long" if the number of characters in the user input for the password is less than 8.

Write another class called **DemoApp** to get the user inputs for the username, and password and then validate those for the above two conditions. If the input has one or more custom exceptions, you should have proper try-catch statements to handle the exceptions.

Sample Output:



```
Enter a username: Kamal
Enter a password: Kamal123
Error: Username must be at least 6 characters long.
```

```
Enter a username: KamalPerera
Enter a password: Ka1
Error: Password must be at least 8 characters long.
```

Question 02

Implement a program that validates user input for item pricing in a Warehouse to ensure the system maintains accurate records and catches any errors in data entry. Write a program to allow the warehouse manager to enter the price of an item. The program should validate the input using three custom exception classes as follows:

Create the following custom exceptions to handle different pricing errors:

- **NegativePriceException:** If the entered price is less than zero, throw this exception. Display the error message as "Price cannot be negative. Please enter a valid amount.".
- **PriceOutOfRangeException:** If the price is not within the valid range of $1 to $10,000, throw this exception. Display the error message as "Price out of range. Please enter a value between $1 and $10,000.
- **PriceNotNumericException:** If the entered price is not a numeric value, throw this exception. Display the error message as "Invalid input. Please enter a numeric value for the price."

b) Create another class, **InventoryProcessor**, to prompt the user for item pricing and manage input validation. Use the exception classes to handle errors in the input. Ensure that the program includes appropriate try-catch blocks to handle each exception and provide clear error messages for the user.

Question 03

Develop a Java program for managing a simple to-do list using ArrayList. The program should perform the following tasks:

(a) Create an empty ArrayList to store task descriptions. Prompt the user to enter tasks one by one (each as a String); keep accepting inputs until the user enters "-99". Add each entered task to the ArrayList.

(b) Once the user finishes entering tasks ("-99"), ask the user to enter the **index** of the task that needs to be removed.

- If the index is valid (between 0 and list.size() – 1), remove the corresponding task and display ***"Task removed successfully."***
- If the index is out of bounds, display ***"Invalid index. No task removed."***

[Hint: use list.size() and list.remove(int index)]

(c) Find and display the total number of tasks currently in the ArrayList.

(d) Print all the remaining tasks in the ArrayList, one per line.

Question 04

Develop a Java program to manage a **Library Inventory** using an ArrayList. The program should perform the following tasks:

(a) Define a **Book** class with two attributes as isbn, and title. Include a constructor to initialize both fields and a method *displayDetails()* that prints the book's ISBN and title.

(b) Implement a class, create an empty ArrayList named inventory. Prompt the user to enter book details one at a time: first the ISBN, then the title. Keep accepting entries until the user enters "-99" as the ISBN. For each valid entry, instantiate a new Book and add it to inventory.

(c) After input is complete, prompt the user to enter the ISBN of the book they wish to remove from the inventory

- If a Book with that ISBN exists in inventory, remove it and display ***Book removed successfully.".***
- If not found, display ***"ISBN not found. No book removed."***

(d) Display the total number of books currently in inventory.

(e) Print all remaining books in the inventory by calling each object's *displayDetails()* method.