National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 1

Time : 2 hours **Max Marks : 20**

1) Create a banking system with the following requirements:

a) Abstract Class BankAccount:

- Attributes: balance, accountNumber.
- Methods:
  - deposit(double amount) (abstract): Deposits money into the account.
  - withdraw(double amount) (abstract): Withdraws money from the account.
  - displayBalance(): Displays the current balance.

b) Subclasses:

- **SavingsAccount:**
  - Implements deposit and withdraw methods.
  - Calculates tax (implement Taxable interface).
- **CurrentAccount:**
  - Implements deposit and withdraw methods.

c) Interface Taxable:

- Method calculateTax(): Calculates tax on balance (implemented by SavingsAccount).

d) Exception Handling:

- Custom exception InsufficientFundsException: Thrown when withdrawal amount exceeds balance.

e) Polymorphism:

- Demonstrate runtime polymorphism by processing different BankAccount objects (Savings, Current).

National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 2

Time : 2 hours **Max Marks : 20**

1) Design a **Vehicle Rental System** that allows users to rent and return vehicles like cars and bikes.

Requirements:

a) Abstract Class Vehicle:

- Define two abstract methods: start() and stop().
- Store the vehicle name.

b) Subclasses (Car and Bike):

- Implement start() and stop() methods for each type of vehicle.

c) Interface Rental:

- Define methods rentVehicle(String vehicleName) and returnVehicle(String vehicleName).

d) Exception Handling:

- Create a VehicleNotAvailableException to handle unavailable vehicles during rental.

e) Class RentalManager:

- Implement the Rental interface.
- Maintain lists for available and rented vehicles.
- Handle rental operations and exceptions.

f) Polymorphism:

- Use polymorphism to treat both Car and Bike as Vehicle objects.

National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 3

Time : 2 hours **Max Marks : 20**

1) Design a Library Management System (LMS) with the following components:

a) Abstract Class Book:

- Methods:
  - borrow(): Borrows the book if available, throws BookNotAvailableException if not.
  - returnBook(): Returns the book and marks it as available.
  - isAvailable(): Checks if the book is available for borrowing.

b) Class Fiction (extends Book):

- Implements the borrow() and returnBook() methods from the Book class.

c) Class NonFiction (extends Book):

- Implements the borrow() and returnBook() methods from the Book class.

d) Exception Handling:

- Create a BookNotAvailableException to handle unavailability during borrowing.

e) Demonstrate Polymorphism:

- Create instances of Fiction and NonFiction.
- Use polymorphism to borrow, return, and demonstrate the availability of different books.
- Handle exceptions for unavailable books.

National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 4

Time : 2 hours **Max Marks : 20**

1) Design an Employee Management System with the following features:

a) Abstract Class Employee:

- Attributes: name, id, salary.
- Methods:
  - calculateSalary(): Abstract method to calculate salary.
  - getDetails(): Abstract method to return employee details.
- Constructor throws an exception InvalidSalaryException for negative salary.

b) Interface Bonus:

- Method calculateBonus(): Calculates bonus as a percentage of salary.

c) Subclasses: Manager and Developer:

- **Manager**:
  - Additional attribute incentive.
  - calculateSalary(): Base salary + incentive.
  - calculateBonus(): 10% of total salary.
- **Developer**:
  - Additional attribute performanceBonus.
  - calculateSalary(): Base salary + performance bonus.
  - calculateBonus(): 15% of total salary.

d) Custom Exception: InvalidSalaryException:

- Thrown when salary is negative.

e) Method processPayroll():

- Accepts an array of Employee objects and processes payroll.
- Displays employee details and bonus.

f) Main Method:

- Create an array of employees (Managers and Developers).
- Process payroll and handle exceptions for invalid salary.

National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 5

Time : 2 hours **Max Marks : 20**

1) Design a system to manage student details, including registration, marks, and scholarship eligibility.

Requirements:

a) Abstract Class - `Student`:
   a. **Attributes**: `name` (String), `id` (int), `marks` (double).
   b. **Methods**:
      i. `register()`: Abstract method for student registration.
      ii. `getMarks()`: Abstract method to return student's marks.

b) Subclass - `Undergraduate`:
   a. Extends `Student`.
   b. Implements `register()` and `getMarks()`.
   c. Implements `checkEligibility()`: Returns `true` if marks >= 75.

c) Subclass - `Postgraduate`:
   a. Extends `Student`.
   b. Implements `register()` and `getMarks()`.
   c. Implements `checkEligibility()`: Returns `true` if marks >= 80.

d) Interface - `ScholarshipEligible`:
   a. Method `checkEligibility()`: Returns `boolean`.

e) Exception - `InvalidMarkException`:
   a. Thrown when marks are out of range (0-100).

f) Polymorphism:
   a. Use `Student` references to manage both `Undergraduate` and `Postgraduate` objects.
   b. Check eligibility using `checkEligibility()`.

National Institute of Technology Kurukshetra
Department of Computer Engineering
Object Oriented Programming using Java (CSPC – 205)
End Term Practical Examination, November 2024
SET - 7

Time : 2 hours **Max Marks : 20**

1) Write a Java program that prints all possible outcomes when a coin is tossed n times. Each toss can result in either "H" (Heads) or "T" (Tails). The program should use recursion to generate all possible combinations of outcomes. The order of outcomes in the output does not matter.

Input:
An integer n representing the number of times the coin is tossed.

Output:
All possible combinations of outcomes, each combination on a new line.

2) Create an abstract class **Person** with fields **name** and **age**, and an **abstract method getRole()**. Create subclasses **Doctor** and **Patient** with additional fields **(specialization for Doctor and disease for Patient)**.
Implement an interface **Billable** in Patient with a method **generateBill()**, which calculates charges based on treatment type using predefined rates:

- Rs.500 for consultation
- Rs.15,000 for surgery.
- Rs.3,000 for therapy
- Rs.1,000 for others.

Use constructor chaining to initialize all fields.