Sukkur IBA University
Merit - Quality - Excellence

CSC-150 - Object Oriented Programming

Semester II (Spring 2023)
Course Instructor: Abdul Haseeb

Objective(s):

Lab 08
Abstraction in Java

1. Abstraction
2. Abstract Classes and Methods
3. Concrete methods vs Abstract Methods

1. Create an abstract class called BankAccount with abstract methods deposit() and withdraw(). Also add instance variable named service_years, after that add a concrete method named show(), which displays inside show method of BankAccountClass. Implement two subclasses, CheckingAccount and SavingsAccount, that inherit from BankAccount and provide their own implementations of the abstract methods. In the main method, create instances of the two subclasses and demonstrate how they can be used to deposit and withdraw funds. Finally Try to call show() method.

2. Create an abstract class called Animal with an abstract method makeSound(), Try to add a static method which displays Animal Class. Implement two subclasses, Dog and Cat, that inherit from Animal and provide their own implementations of the makeSound() method. In the main method, create instances of the two subclasses and demonstrate how they can be used to make different animal sounds.

3. Create an abstract class called Game with an abstract method play(). Implement two subclasses, Chess and Checkers, that inherit from Game and provide their own implementations of the play() method. In the main method, create instances of the two subclasses and demonstrate how they can be used to play different types of games.

4. We have to calculate the percentage of marks obtained in three subjects (each out of 100) by student A and in four subjects (each out of 100) by student B. Create an abstract class 'Marks' with an abstract method 'getPercentage'. It is inherited by two other classes 'A' and 'B' each having a method with the same name which returns the percentage of the students. The constructor of student A takes the marks in three subjects as its parameters and the marks in four subjects as its parameters for student B. Create an object for each of the two classes and print the percentage of marks for both the students.

Thanos Task

The following class diagram represents a library system that records the books that have been borrowed. There are three kinds of books: regular books, reference books, and audio books.

Reference books can be taken out for just two days, while other kinds of books may be borrowed for two weeks. The overdue fees are 10 cents per day for reference books and regular books, and 20 cents per day for audio books.

Audio books and regular books have both authors and titles; reference books only have titles.

The day when the book is taken out and the day due are counted as days since the library opened on New Year's Day in 2001. So, for example, an audio book taken out recently would be recorded as taken out on the day 6371 with due date on the day 6385.

```
                  +--------+
                  | IBook  |
                  +--------+
                     / \
                     ---
                      |
       ---------------------------------------
       |                  |                  |
+----------------+ +----------------+ +----------------+
| Book           | | RefBook        | | AudioBook      |
+----------------+ +----------------+ +----------------+
| String title   | | String title   | | String title   |
| String author  | | int dayTaken   | | String author  |
| int dayTaken   | +----------------+ | int dayTaken   |
+----------------+                    +----------------+
```

- Design the interfaces and classes that represent the library borrowing system.
- Define the abstract class ABook and lift those fields that can be lifted to this class.
- Design the method daysOverdue that consumes the number that represents today in the library date-recording system and produces the number of days this book is overdue. If the number is negative, the book can still be out for that many days.
- Design the method isOverdue that produces a boolean value that informs us whether the book is overdue on the given day.
- Design the method computeFine that computes the fine for this book, if the book is returned on the given day.

For all methods, think carefully whether they should be designed being implemented solely in the abstract class, implemented solely in the concrete classes, or implemented in the abstract class and then overridden in some of the concrete classes.