by Kunal Sir

Assignment:Abstraction (abstract class)

Part A: MCQ's

1. Abstract class in Java is also known as:
A. Fully implemented class
B. Partial class
C. Concrete class
D. Final class

2. If a class contains at least one abstract method, then the class must be:
A. Final
B. Static
C. Abstract
D. Concrete

3. Which type of methods can an abstract class contain?
A. Only abstract methods
B. Only concrete methods
C. Both abstract and concrete methods
D. Only static methods

4. Can an abstract class have all implemented methods and still be abstract?
A. No
B. Yes
C. Only in Java 8
D. Only if static methods exist

5. Can we create an object of an abstract class?
A. Yes
B. No
C. Only using inheritance
D. Only inside same package

6. Which statement is TRUE about abstract class reference?
A. Reference cannot be created
B. Reference can refer only abstract class object
C. Reference can refer to subclass object
D. Reference can refer to interface object

7. Which keyword is used to inherit an abstract class?
A. implements
B. extends
C. inherit
D. super

8. Abstract class supports which type of variables?
A. Only static final
B. Only non-static
C. Final, non-final, static, non-static
D. Only final

9. Which inheritance is NOT supported by abstract class?
A. Single inheritance
B. Hierarchical inheritance
C. Multiple inheritance
D. Multilevel inheritance

10. Which of the following cannot be declared in an abstract class?
A. Constructor
B. Static method
C. Final method
D. Abstract method with body

11. Abstract keyword is used to:
A. Increase performance
B. Create concrete class
C. Achieve abstraction
D. Support multiple inheritance

12. Which statement correctly differentiates abstract class and interface?
A. Abstract class supports multiple inheritance
B. Interface supports concrete methods
C. Abstract class supports concrete methods
D. Interface supports instance variables

13. Can an abstract class be declared as final?
A. Yes
B. No
C. Only if no abstract methods
D. Only in Java 11

14. Abstract methods are:
A. Methods with body
B. Methods without body
C. Static methods
D. Private methods

15. Which class can contain abstract methods?
A. Final class
B. Concrete class
C. Abstract class
D. Object class

16. Abstract class helps in:
A. Data hiding only
B. Code duplication
C. Reusability and abstraction
D. Memory management

17. If subclass does not implement all abstract methods, then subclass must be:
A. Concrete
B. Static
C. Abstract
D. Final

18. Which example best represents abstract class?
A. RBI rules
B. ATM machine
C. Transportation system
D. Marker interface

19. Abstract class constructor is used to:
A. Create object
B. Initialize static variables
C. Initialize common data
D. Prevent inheritance

20. Abstract class is preferred over interface when:
A. Multiple inheritance is required
B. Only abstract methods are needed
C. Some common implementation is required
D. No method implementation is needed

Part B: Problem Statements

1. Transportation System

Problem Statement

In real life, all vehicles are used for transportation, but not all vehicles move in the same way. A car moves on roads, a train runs on tracks, and a ship moves on water. However, all vehicles share some common behavior like using fuel and starting the journey.

To represent this situation in programming, create an abstract class `Transport` that contains common methods such as `fuelType()` and an abstract method `move()`. Each vehicle class must provide its own implementation of the `move()` method.

Sample Input

Vehicle selected: Car

Sample Output

```
Fuel type: Petrol
Car is moving on the road
```

2. Bank Account System

Problem Statement

In a bank, different types of accounts exist such as Savings Account and Current Account. All accounts allow deposit and withdrawal operations, but the way interest is calculated differs for each account type.

Create an abstract class `BankAccount` that contains common methods like `deposit()` and an abstract method `calculateInterest()`. Each account type implements its own interest calculation logic.

Sample Input

Account Type: Savings
Balance: 10000

Sample Output

```
Amount deposited successfully
Savings account interest calculated
```

---

3. Employee Management System

Problem Statement

An organization has different types of employees such as full-time and part-time employees. All employees have basic details like name and ID, but salary calculation depends on employee type.

Create an abstract class `Employee` that contains common employee details and an abstract method `calculateSalary()`. Each employee type must define its own salary calculation logic.

Sample Input

Employee Type: Part-Time
Working Hours: 20

Sample Output

```
Employee details displayed
Salary calculated: 4000
```

---

4. Shape Area Calculator

Problem Statement

Different shapes like circle, rectangle, and triangle have different formulas to calculate area, but every shape must have an area.

Create an abstract class `Shape` that declares an abstract method `area()` and a concrete method `display()`. Each shape class provides its own formula for calculating area.

Sample Input

Shape: Circle
Radius: 7

Sample Output

```
Shape displayed
Area of Circle: 153.94
```

5. Payment Processing System

Problem Statement

An online shopping application allows customers to pay using UPI, credit card, or net banking. Every payment needs to be validated first, but the actual payment process differs for each method.

Create an abstract class `Payment` that contains a concrete method `validatePayment()` and an abstract method `pay()`.

Sample Input

Payment Method: UPI
Amount: 500

Sample Output

```
Payment validated
Payment of 500 completed using UPI
```

---

6. Media Player System

Problem Statement

A media player application can play audio files and video files. While stopping the media is common, the playing mechanism differs.

Create an abstract class `MediaPlayer` with a concrete method `stop()` and an abstract method `play()`.

Sample Input

Media Type: Audio

Sample Output

```
Playing audio file
Media stopped
```

---

7. Vehicle Rental System

Problem Statement

A vehicle rental company rents bikes and cars. The pricing logic is common, but the rental process differs for each vehicle.

Create an abstract class `Vehicle` with common rental details and an abstract method `rentVehicle()`.

Sample Input

Vehicle Type: Bike
Rental Days: 3

Sample Output

```
Bike rented for 3 days
Rent calculated successfully
```

---

8. Online Shopping Product System

Problem Statement

An online shopping platform sells different types of products such as electronics and clothing. All products have price and description, but final pricing differs based on product type.

Create an abstract class `Product` with common product information and an abstract method `calculateFinalPrice()`.

Sample Input

Product Type: Electronics
Base Price: 20000

Sample Output

```
Product details displayed
Final price calculated
```

9. Bank Loan System

Problem Statement

Banks provide different types of loans such as home loans and car loans. Loan processing steps are common, but interest rates differ.

Create an abstract class `Loan` that contains common processing logic and an abstract method `calculateInterest()`.

Sample Input

Loan Type: Home Loan
Loan Amount: 10,00,000

Sample Output

```
Loan details displayed
Home loan interest applied
```

---

10. User Authentication System

Problem Statement

A system allows users to log in using password or OTP. The authentication process is similar, but verification logic differs.

Create an abstract class `Authenticator` that defines the authentication structure.

Sample Input

Login Method: OTP

Sample Output

```
Authentication started
User authenticated using OTP
```
