1. Write a Java class `Student` that has multiple constructors. One constructor should accept student name and ID, while another should accept student name, ID, and grades. Implement constructor overloading and ensure the constructors call each other using `this()`.

2. Create a class `Employee` with fields `name`, `id`, `designation`, and `salary`. Implement multiple constructors that initialize different combinations of these fields, and ensure that they chain to a primary constructor using the `this()` keyword.

3. Design a class `ComplexNumber` that models complex numbers. Write a copy constructor that takes another `ComplexNumber` object and initializes the current object's real and imaginary parts with the copied values.

4. Implement a singleton class `DatabaseConnection` using a private constructor. Ensure that the class restricts object creation to only one instance and provides a global access point using a static method.

5. Write an abstract class `Shape` that has a parameterized constructor to initialize the color of the shape. Extend this class in `Circle` and `Rectangle`, which will have their own additional parameters (like radius, length, width). Ensure proper constructor calls using `super()`.

6. Implement a class `Polynomial` that models a polynomial equation. Use a constructor that takes a variable number of coefficients (using varargs) and initializes the polynomial. Write a method to display the polynomial in a readable format (e.g., `3x^2 + 2x + 1`).

7. Create an interface `Shape` that has methods `double area()` and `double perimeter()` . Implement this interface in two classes: `Circle` and `Rectangle`. The `Circle` class should calculate the area and perimeter using the radius, and the `Rectangle` class should use the length and width.

8. Create two interfaces `Flying` and `Swimming`, each with a method `void fly()` and `void swim()` respectively. Create a class `Duck` that implements both interfaces and overrides both methods. Write a test class that demonstrates the `Duck`'s ability to both fly and swim.

9. Create an interface `PaymentMethod` with a method `void pay(double amount)`. Implement this interface in two classes: `CreditCardPayment` and `PayPalPayment`. Write a class `OnlineStore` that accepts a `PaymentMethod` in its constructor and uses it to process a payment.