Polymorphism Question Set (20 Questions)

Section 1: Basic Level (5 Questions)

1. Define Polymorphism in Java and explain its types.
2. Write a method overloading example with two methods named `add`.
3. Explain the difference between compile-time and runtime polymorphism.
4. Write a program where a child class overrides a parent class method using `@Override`.
5. Create an abstract class `Shape` with an abstract method `area()` and implement it in a subclass `Circle`.

Section 2: Easy-to-Hard Level (5 Questions)

1. Write a method overloading example that demonstrates overloading with different parameter types and numbers.
2. Implement method overriding with covariant return type.
3. Create an abstract class `Animal` with concrete method `sleep()` and abstract method `sound()`. Implement it in two child classes.
4. Explain `@Override` annotation with an example where it prevents compile-time error.
5. Write a program using an array of `Animal` objects to demonstrate runtime polymorphism.

Section 3: Medium-to-Hard Level (5 Questions)

1. Implement polymorphism using `ArrayList` to store multiple child class objects.
2. Explain the advantages of using collections over arrays in polymorphism.
3. Create an example showing how parent class reference can hold multiple child objects and invoke overridden methods.
4. Write a program that demonstrates method overloading with type promotion.
5. Create a class hierarchy where multiple levels of inheritance exist and show runtime polymorphism.

Section 4: Intermediate-to-Hard Level (5 Questions)

1. Implement a program where child class overrides a method and changes the return type to a subclass (covariant return type), then demonstrate dynamic dispatch.
2. Create an abstract class `Vehicle` with methods `start()` and `stop()`. Implement multiple child classes and store them in an array to demonstrate polymorphism.
3. Write a program that uses both method overloading and method overriding in the same class hierarchy.
4. Create an `ArrayList` of `Shape` objects where `Circle` and `Rectangle` are subclasses. Use polymorphism to calculate total area.
5. Implement multiple child classes of an abstract class `Employee`, override a method `calculateSalary()`, and demonstrate polymorphism using collection of objects.