Here are some exercises to practice inheritance concepts, specifically, overriding, overloading, abstract classes, constructor chaining and polymorphism.

### Inheritance 1

```java
class Vehicle {
    public void drive() {
        System.out.println("drivedrive");
    }
}

class Car extends Vehicle {
}
class Bike extends Vehicle {
}

class Main {
    public static void main(String[] args) {
        Car c = new Car();
        c.drive();
        Bike b = new Bike();
        b.drive();
    }
}
```

### Inheritance 2

```java
class Vehicle {
    public void drive() {
        System.out.println("drivedrive");
    }
}

class Car extends Vehicle {
    public void drive() {
        System.out.println("rollroll");
    }
}
class Bike extends Vehicle {
    public void drive() {
        System.out.println("pedalpedal");
    }
}

class Main {
```

If it compiles, what does it print?
public static void main(String[] args) {
    Car c = new Car();
    c.drive();
    Bike b = new Bike();
    b.drive();
}

Inheritance 3

class Vehicle {
    public void drive() {
        System.out.println("drivedrive");
    }
}
class Car extends Vehicle {
    public void drive() {
        super.drive();
    }
}
class Bike extends Vehicle {
    public void drive() {
        System.out.println("pedalpedal");
    }
}
class Main {
    public static void main(String[] args) {
        Car c = new Car();
        c.drive();
        Bike b = new Bike();
        b.drive();
    }
}

Inheritance 4

abstract class Vehicle {
    public void drive() {
        System.out.println("drivedrive");
    }
}
class Car extends Vehicle {
    public void drive() {
        System.out.println("rollroll");
    }
}
class Bike extends Vehicle {
    public void drive() {
        System.out.println("pedalpedal");
    }
}
Inheritance 5

abstract class Vehicle {
    public abstract void drive();
}

class Car extends Vehicle {
}
class Bike extends Vehicle {

class Main {
    public static void main(String[] args) {
        Car c = new Car();
        c.drive();
        Bike b = new Bike();
        b.drive();
    }
}

Inheritance 6

abstract class Vehicle {
    public void drive() {
        System.out.println("drivedrive");
    }
}

class Car extends Vehicle {
    public void drive() {
        System.out.println("rollroll");
    }
}
class Bike extends Vehicle {
    public void drive() {
        System.out.println("pedalpedal");
    }
}
class Main {
    public static void main(String[] args) {
        Vehicle c = new Car();
        c.drive();
        Vehicle b = new Bike();
        b.drive();
    }
}
abstract class Vehicle {
    private void drive() {
        System.out.println("drivedrive");
    }
}

class Car extends Vehicle {
    public void drive() {
        System.out.println("rollroll");
    }
}
class Bike extends Vehicle {
    public void drive() {
        System.out.println("pedalpedal");
    }
}

class Main {
    public static void main(String[] args) {
        Car c = new Car();
        c.drive();
        Vehicle b = new Bike();
        b.drive();
    }
}

Inheritance 8

abstract class Vehicle {
    public void drive() {
        System.out.println("drivedrive");
    }
}

class Car extends Vehicle {
    public void drive(int count) {
        System.out.println("rollroll " + count);
    }
}
class Bike extends Vehicle {
    public void drive(int count) {
        System.out.println("pedalpedal " + count);
    }
}

class Main {
    public static void main(String[] args) {
        Car c = new Car();
        c.drive(10);
Vehicle b = new Bike();
b.drive(5);
}
}

Overloading 1

class Addition{
    public int add(int a, int b){
        int sum = a+b;
        return sum;
    }

    public int add(int a, int b, int c){
        int sum = a+b+c;
        return sum;
    }

    public double add(double a, double b, double c){
        double sum = a+b+c;
        return sum;
    }
}

class Main {
    public static void main (String[] args) {
        Addition ob = new Addition();

        int sum1 = ob.add(1,2);
        System.out.println(sum1);
        int sum2 = ob.add(1,2,3);
        System.out.println(sum2);
        double sum3 = ob.add(1.0,2.0,3.0);
        System.out.println(sum3);
    }
}

Overloading 2

class Birthday {
    public void greet(String name, int age){
        System.out.println("Happy " + age + ". birthday, " + name + "!");
    }

    public void greet(int age, String name){
        System.out.println("All the best for your " + age + ". birthday, " + name + "!");
    }
}

class Main {
    public static void main (String[] args) {
        Birthday b = new Birthday();
        b.greet("Jack", 5);
    }
}
b.greet(7, "Jill");

Overloading 3

class Addition{
    public int add(int a, int b){
        int sum = a+b;
        return sum;
    }
    public double add(int a, int b){
        int sum = a+b;
        return sum;
    }
}

class Main {
    public static void main (String[] args) {
        Addition ob = new Addition();
        int sum1 = ob.add(1,2);
        System.out.println(sum1);
        double sum2 = ob.add(1,2);
        System.out.println(sum2);
    }
}

Inheritance 9

abstract class Vehicle {
    private String noise;

    public Vehicle() {
        noise = "drive";
    }

    public void drive() {
        System.out.println(noise + noise);
    }
}

class Car extends Vehicle {}

class Bike extends Vehicle {}

class Main {
    public static void main(String[] args) {
        Vehicle c = new Car();
        c.drive();
        Vehicle b = new Bike();
        b.drive();
    }
}
abstract class Vehicle {
    private String noise;

    public Vehicle() {
        noise = "drive";
    }

    public void drive() {
        System.out.println(noise + noise);
    }
}

class Car extends Vehicle {
    public void drive() {
        System.out.println(noise);
    }
}

class Bike extends Vehicle {
    public void drive() {
        System.out.println("pedal" + noise);
    }
}

class Main {
    public static void main(String[] args) {
        Vehicle c = new Car();
        c.drive();
        Vehicle b = new Bike();
        b.drive();
    }
}

Inheritance 11

abstract class Vehicle {
    protected String noise;

    public Vehicle() {
        noise = "drive";
    }

    public void drive() {
        System.out.println(noise + noise);
    }
}

class Car extends Vehicle {
    public void drive() {
        System.out.println(noise);
    }
}
class Bike extends Vehicle {
    public void drive() {
        System.out.println("pedal" + noise);
    }
}

class Main {
    public static void main(String[] args) {
        Vehicle c = new Car();
        c.drive();
        Vehicle b = new Bike();
        b.drive();
    }
}

Inheritance 12

abstract class Vehicle {
    protected String noise;

    public Vehicle(String noise) {
        this.noise = noise;
    }

    public void drive() {
        System.out.println(noise + noise);
    }
}

class Car extends Vehicle {
    public void drive() {
        System.out.println(noise);
    }
}

class Bike extends Vehicle {
    public void drive() {
        System.out.println("pedal" + noise);
    }
}

class Main {
    public static void main(String[] args) {
        Vehicle c = new Car("roll");
        c.drive();
        Vehicle b = new Bike();
        b.drive();
    }
}

Inheritance 13

abstract class Vehicle {
protected String noise;

public Vehicle(String noise) {
    this.noise = noise
}

g public void drive() {
    System.out.println(noise + noise);
}
}

class Car extends Vehicle {
    public Car(String noise) {
        this.noise = noise;
    }

    public void drive() {
        System.out.println(noise);
    }
}

class Bike extends Vehicle {
    public Bike(String noise) {
        this.noise = noise;
    }

    public void drive() {
        System.out.println(noise + noise + noise);
    }
}

class Main {
    public static void main(String[] args) {
        Vehicle c = new Car("roll");
        c.drive();
        Vehicle b = new Bike("pedal");
        b.drive();
    }
}

Inheritance 14

abstract class Vehicle {
    protected String noise;

    public Vehicle(String noise) {
        this.noise = noise;
    }

    public void drive() {
        System.out.println(noise + noise);
    }
}

class Car extends Vehicle {

public Car(String noise) {
    super(noise);
}

public void drive() {
    System.out.println(noise);
}
}
class Bike extends Vehicle {
    public Bike(String noise) {
        super(noise);
        this.noise = "zoom";
    }

    public void drive() {
        System.out.println(noise + noise + noise);
    }
}

class Main {
    public static void main(String[] args) {
        Vehicle c = new Car("roll");
        c.drive();
        Vehicle b = new Bike("pedal");
        b.drive();
    }
}

Solutions

Inheritance

1. Compiles and prints "drivedrive" twice because Bike and Car inherit drive from Vehicle
2. Compiles and prints "rollroll" and "pedalpedal" because Bike and Car override drive
3. Compiles and prints "drivedrive" and "pedalpedal" because Car's drive method calls Vehicle's drive method.
4. Does not compile because abstract classes cannot be instantiated.
5. Does not compile because abstract methods must be implemented / overridden in the subclass.
6. Compiles and prints "rollroll" and "pedalpedal" because polymorphism allows subclass instances to be referenced by supertype variables.
7. Does not compile because drive is called on a variable of type Vehicle which has no public access for its method drive. Even though it is overridden by the created instance Bike, the API visible at runtime is the one from the variables type. The execution of c.drive() compiles and would print "rollroll". Access modifiers in overridden methods need to be at least as visible as in their superclass but can have higher visibility in a subclass.
8. Does not compile because the method drive is not overridden but overloaded due to the change in the parameter list. This is allowed and the classes Vehicle, Car and Bike compile just fine. Also, the call c.drive(10) in the Main class is fine. However, the call b.drive(5) does not work because b is of type Vehicle. Vehicle does not have access to a method that was overloaded in one of its subclasses even if it is referencing the Bike subclass under the hood.

9. Prints "drive" twice because the super classes constructor is called after the base classes constructor, drive is inherited by Car and Bike and the inherited version has access to the private member of Vehicle.

10. Does not compile because subclasses don't have access to private members of superclass.

11. Compiles because access to protected members is allowed in subclasses. It prints "drive" and "pedaldrive".

12. Does not compile because subclass Car has no one argument constructor. Constructors are not inherited!

13. Does not compile because subclasses don't call the superclasses' one argument constructor. The constructor chain must be maintained throughout the class hierarchy.

14. Compiles and prints "roll" and "zoomzoomzoom" because the superclass constructors are executed before the subclass constructors, hence the super() call needs to be the very first line in a subclass constructor.

Overloading

1. Compiles and prints 3, 6 and 6.00000 because add is overloaded once by using more parameters and once by using different parameter types.

2. Compiles and prints "Happy 5. birthday, Jack!" and "All the best for your 7. birthday, Jill!" because greet is overloaded by swapping parameter types around.

3. Does not compile because only changing the return type is not enough for overloading.