Spring 2021

CS-240: Object-oriented Programming

Lab-11 Manual

Inheritance

GIFT School of Engineering and Applied Sciences

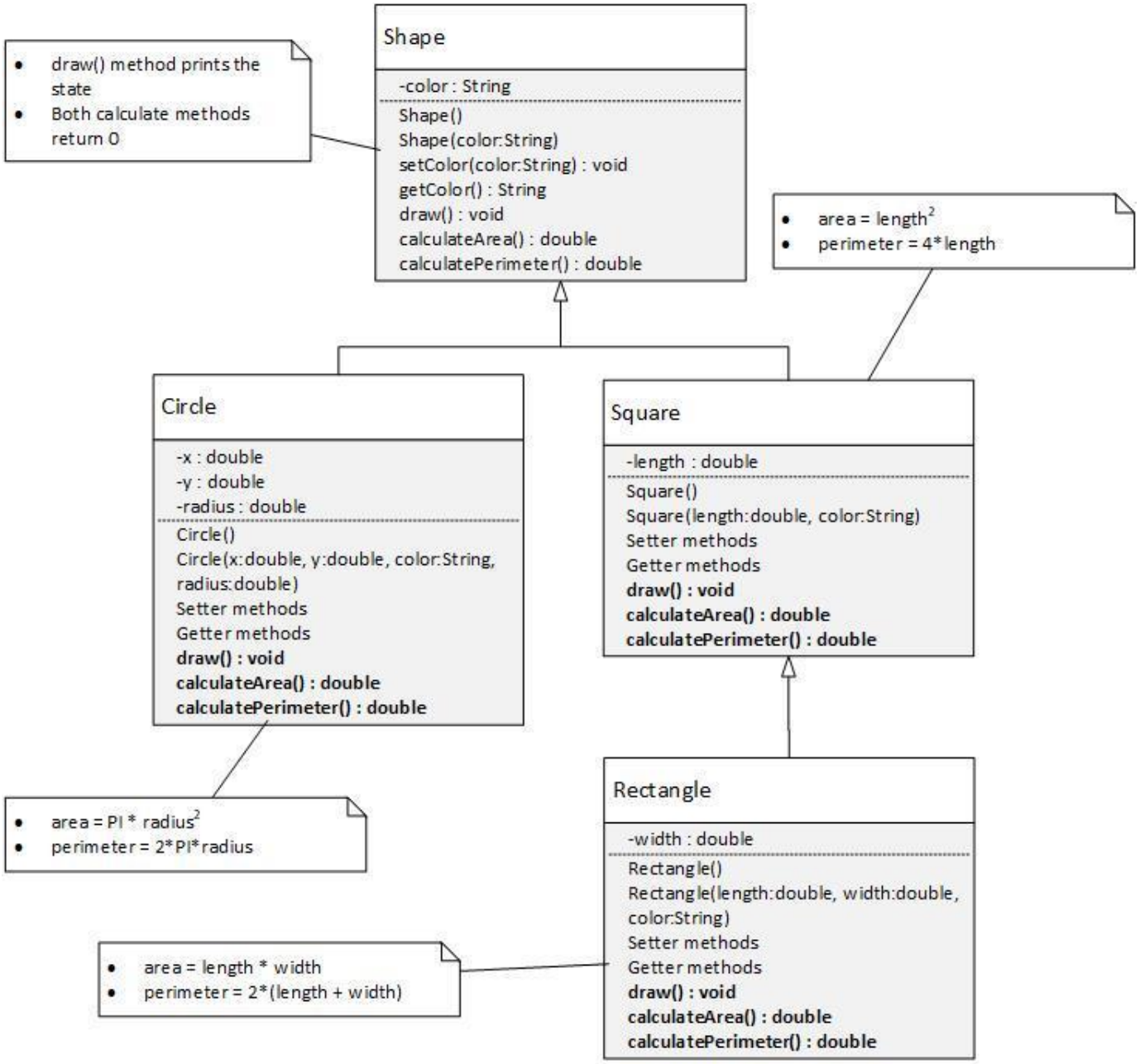
v1.3
3/30/2020

Task #1: Creating Super and Subclasses

In this task, you are being asked to create super and subclasses in Java. You will also be writing code that overrides methods and calls the superclass constructors and methods from subclasses.

NOTE: Write your classes and the *main* method in separate files.

Convert the given UML diagram into classes depicting the inheritance hierarchy shown. Note that **bold methods represent overridden methods** in the below UML diagram.



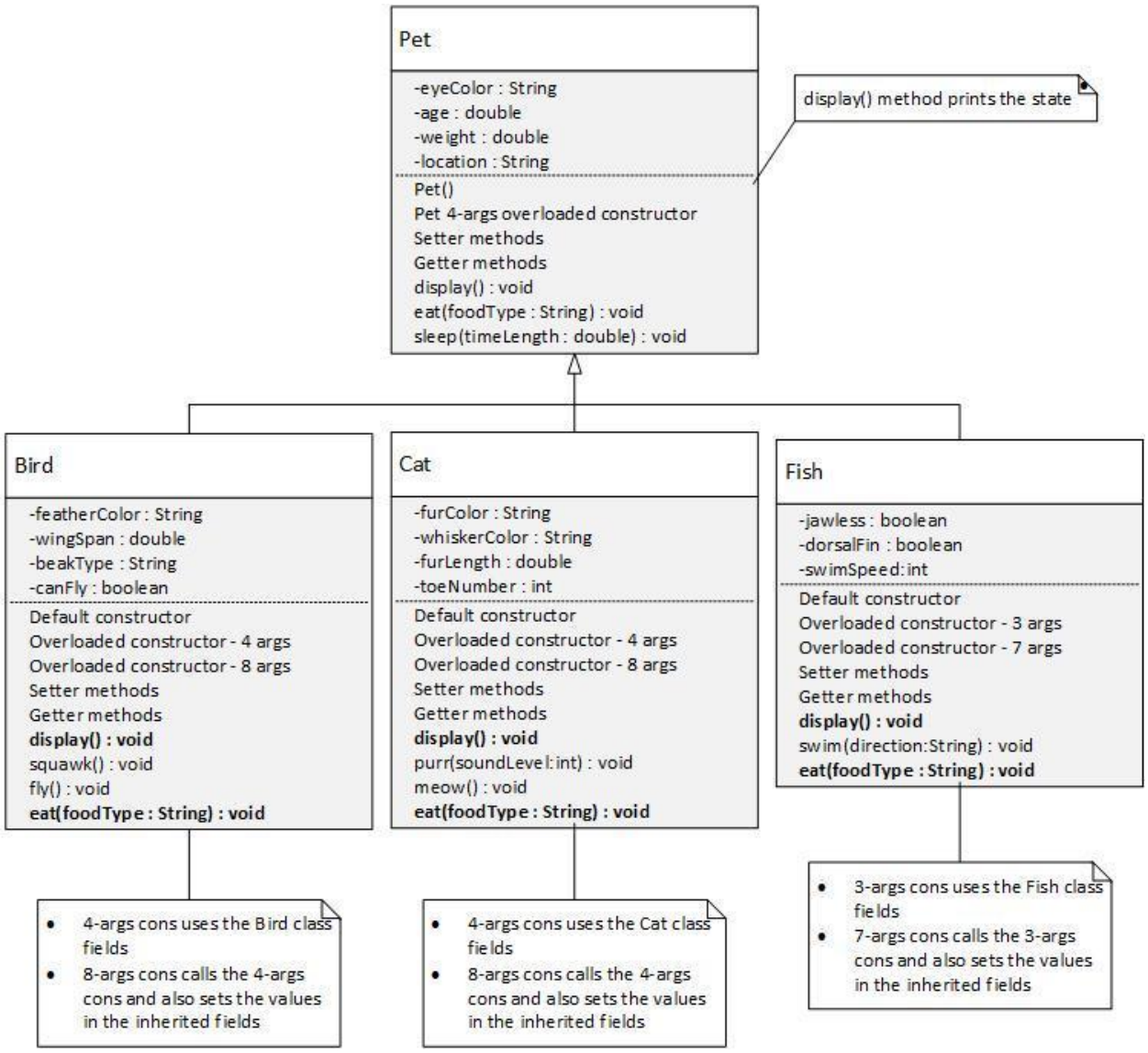
1. Create all classes having names as per the above diagram, and **`UsingShapes.java`** having the **`main`** method**.**
2. Inside the **`main`** method, create instances of **`Circle`, `Square`** and **`Rectangle`** using the default and overloaded constructors. Assign appropriate values to the state of all instances.
3. Display the state of all objects.
4. Display the area and the perimeter of all objects.
5. Use the setters and change the state of few objects.
6. Again, display the area and the perimeter of all objects whose state was changed.

Task #2: Creating Super and Subclasses

In this task, you are being asked to create super and subclasses in Java. You will also be writing code that overrides methods and calls the superclass constructors and methods from subclasses.

NOTE: Write your classes and the *main* method in separate files.

Convert the given UML diagram into classes depicting the inheritance hierarchy shown. Note that **bold methods represent overridden methods** in the below UML diagram.



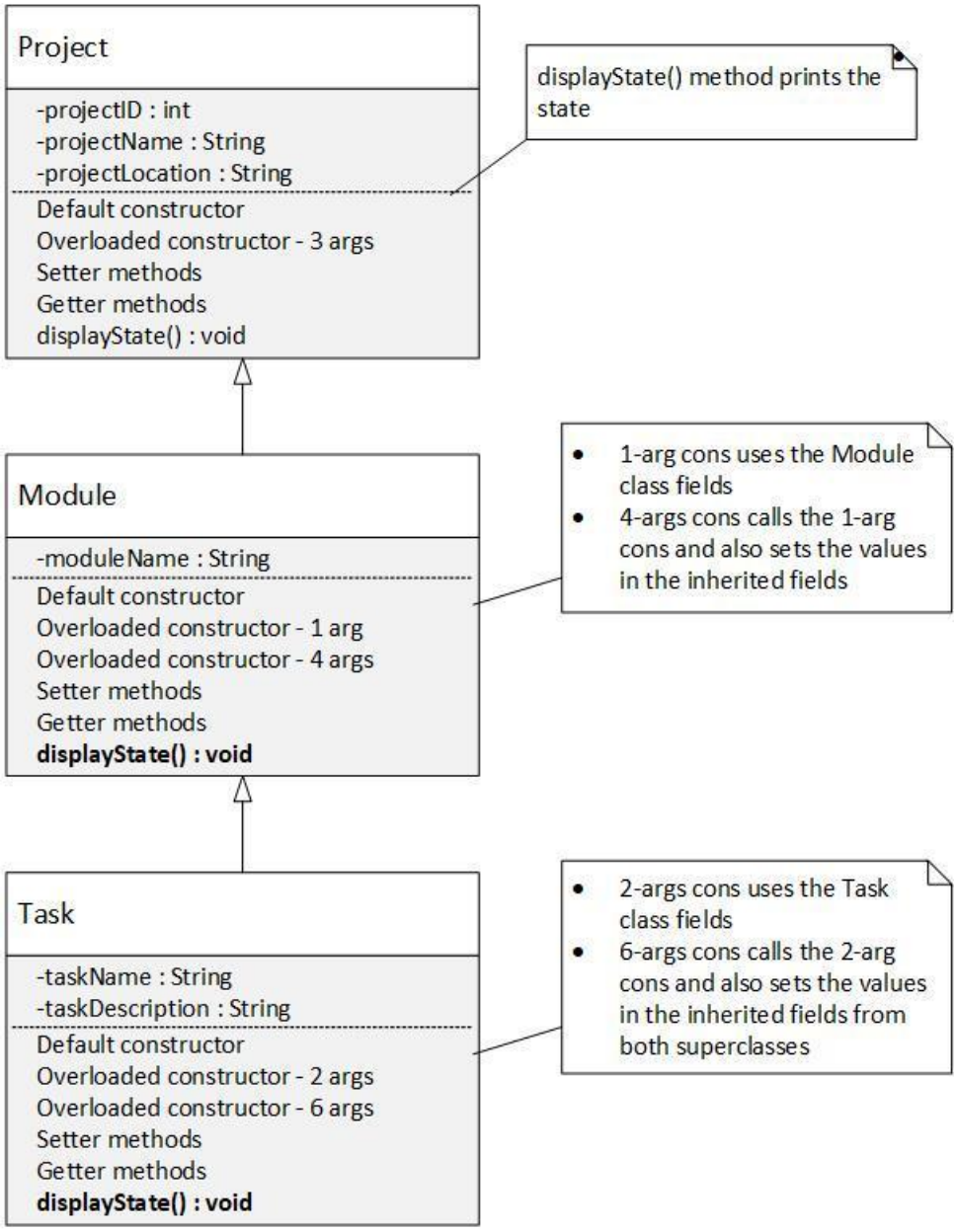
1. Create all classes having names as per the above diagram, and **`UsingPets.java`** having the **`main`** method**.**
2. Inside the **`main`** method, create instances of **`Bird`, `Cat`** and **`Fish`** using the default and overloaded constructors. Assign appropriate values to the state of all instances.
3. Display the state of all objects.
4. Use the setters and change the state of few objects.
5. Again, display the area and the perimeter of all objects whose state was changed.
6. Also, call each method that is unique to that class for each object.

Task #3: Creating Super and Subclasses

In this task, you are being asked to create super and subclasses in Java. You will also be writing code that overrides methods and calls the superclass constructors and methods from subclasses.

NOTE: Write your classes and the *main* method in separate files.

Convert the given UML diagram into classes depicting the inheritance hierarchy shown. Note that **bold methods represent overridden methods** in the below UML diagram.



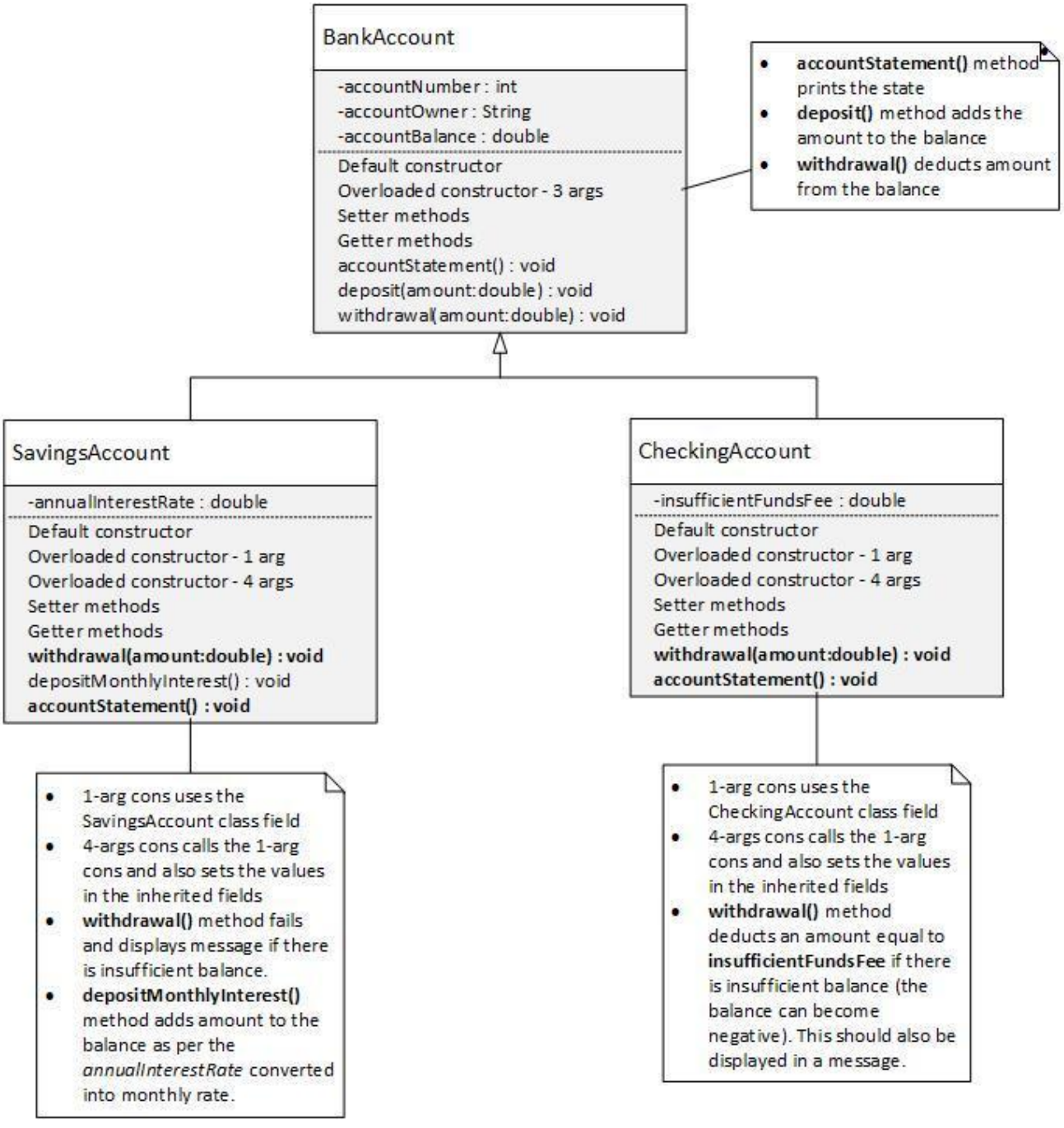
1. Create all classes having names as per the above diagram, and **`UsingTasks.java`** having the **`main`** method**.**
2. Inside the **`main`** method, create instances of **`Module`** and **`Task`** using the default and overloaded constructors. Assign appropriate values to the state of all instances.
3. Display the state of all objects.
4. Use the setters and change the state of few objects.
5. Again, display the state of all objects whose state was changed.

Task #4: Creating Super and Subclasses

In this task, you are being asked to create super and subclasses in Java. You will also be writing code that overrides methods and calls the superclass constructors and methods from subclasses.

NOTE: Write your classes and the *main* method in separate files.

Convert the given UML diagram into classes depicting the inheritance hierarchy shown. Note that **bold methods represent overridden methods** in the UML diagram below.



1. Create all classes having names as per the above diagram, and **`UsingAccounts.java`** having the **`main`** method**.**
2. Inside the **`main`** method, create instances of **`SavingsAccount`** and **`CheckingAccount`** using the default and overloaded constructors. Assign appropriate values to the state of all instances.
3. Display the state of all objects.
4. Apply the **`deposit`** and **`withdrawal`** methods to both type of objects
5. Make sure that the **`CheckingAccount`** balance becomes negative by applying the **`withdrawal`** method multiple times.
6. Apply **`deposit`** method and add amounts to the **`CheckingAccount`** object so that the balance becomes positive again.
7. Apply the **`depositMonthlyInterest`** method to the **`SavingsAccount`** object multiple times.
8. Display the state of all objects.