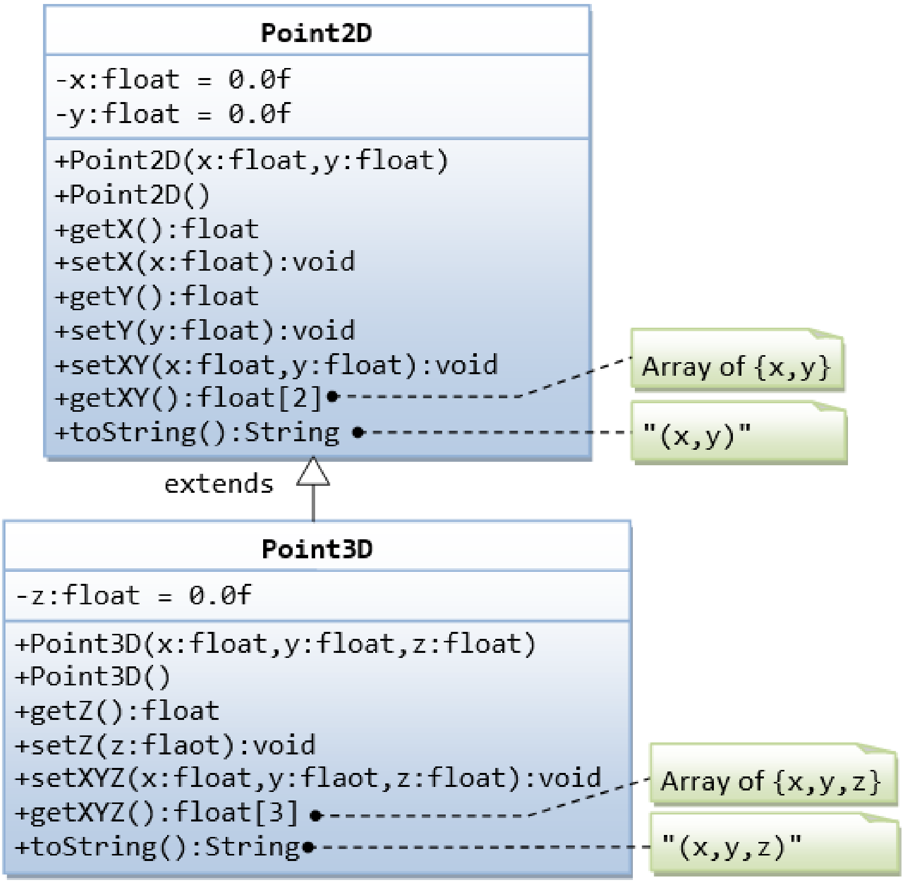
Problem 1 [10 pts] Write the classes as shown in the following class diagram. Mark all the overridden methods with annotation `@Override`.



Problem 2

Examine the following codes and draw the class diagram.

```
abstract public class Animal {
   abstract public void greeting();
}
public class Cat extends Animal {
   @Override
   public void greeting() {
      System.out.println("Meow!");
   }
}
public class Dog extends Animal {
   @Override
   public void greeting() {
      System.out.println("Woof!");
   }
   public void greeting(Dog another) {
      System.out.println("Woooooooooof!");
   }
}
public class BigDog extends Dog {
   @Override
   public void greeting() {
      System.out.println("Woow!");
   }

   @Override
   public void greeting(Dog another) {
      System.out.println("Woooooowwwww!");
   }
}
```

Explain the outputs (or error) for the following test program.

```
public class TestAnimal {
   public static void main(String[] args) {
      // Using the subclasses
      Cat cat1 = new Cat();
      cat1.greeting();                   //Output:
      Dog dog1 = new Dog();
      dog1.greeting();                   //Output:
      BigDog bigDog1 = new BigDog();
      bigDog1.greeting();                //Output:

      // Using Polymorphism
      Animal animal1 = new Cat();        //Would this work?
      animal1.greeting();                //Output:
      Animal animal2 = new Dog();        //Would this work?
      animal2.greeting();                //Output:
      Animal animal3 = new BigDog();     //Would this work?
      animal3.greeting();                //Output:
      Animal animal4 = new Animal();

      // Downcast
      Dog dog2 = (Dog)animal2;           //Would this work?
      BigDog bigDog2 = (BigDog)animal3;  //Would this work?
      Dog dog3 = (Dog)animal3;           //Would this work?
      Cat cat2 = (Cat)animal2;           //Would this work?
      dog2.greeting(dog3);               //Output:
      dog3.greeting(dog2);               //Output:
      dog2.greeting(bigDog2);            //Output:
      bigDog2.greeting(dog2);            //Output:
      bigDog2.greeting(bigDog1);         //Output:
   }
}
```

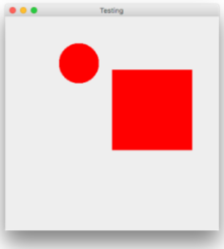
Problem 3 (6 pts) Write a class Animals which can potentially have many animals. Write functions to add and remove animals. Using polymorphism, write code to insert 2 cats, 1 dog, and 1 big dog using your class.

Problem 4 (8 pts) Modify `Animal`.java so that is an interface.

(2 pts) What changes will you make to the subclasses?

Problem 5 (8 pts) Modify Dog.java so that the `greeting` method is an abstract method. (2 pts) What changes will you make to the subclasses?

Task 6 What code will you write to get the following figure? Assume the circle is centered at (200,100) and has a radius of 50 and that the rectangle top left coordinate is at (250, 120) and has a side length of 200.



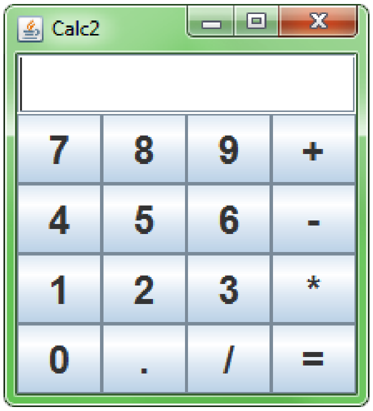
Assume that you are writing inside the class extending JPanel.

```
public void paintComponent(Graphics g)
{




}
```

Task 7 [10 + 5 points] (a) Write code to make the following GUI for a calculator:



Hint: You can have a JPanel inside another JPanel. The parent JPanel can have the JTextField at North and the child JPanel could be added in South. The child JPanel could use a 4 by 4 Grid layout.

(b) Write code for "7" button so that when it is clicked, it appends 7 to what is currently on the screen.

Task 8 [10 points] Write a recursive function that takes a number n and an array of integers as input and the sum of the list. You are not allowed to use the any for or while loop and it should be recursive.

Task 9 [6 points] Given an array of integers, and a target sum, output true of a subset of the integers add up to the sum, false otherwise. Solve this problem using recursion.

Examples:

[3, 5, 7, 11, 13], 15 -> true

[3, 5, 7, 11, 13], -> false

[3, 5, 7, 11, 13], 23 -> true

[3, 5, 7, 11, 13], 40 -> false

Task 10 [15 points] Write a program to find the frequencies of each word in a given file using a HashMap.

Your pseudocode could be:

1. Open the file
2. Create a Hashmap
3. Iterate over the string for each and every word and check whether it is present in the hashmap or not.
    a. If it is not present, insert it in the hashmap with the key as the word and 1 as the value
    b. If it is present, then update the value by 1.
4. You can prompt the user to enter the word. Check if it is present in the hashmap or not.
    a. If it is not present, print out 0
    b. If it is present, then print out the value