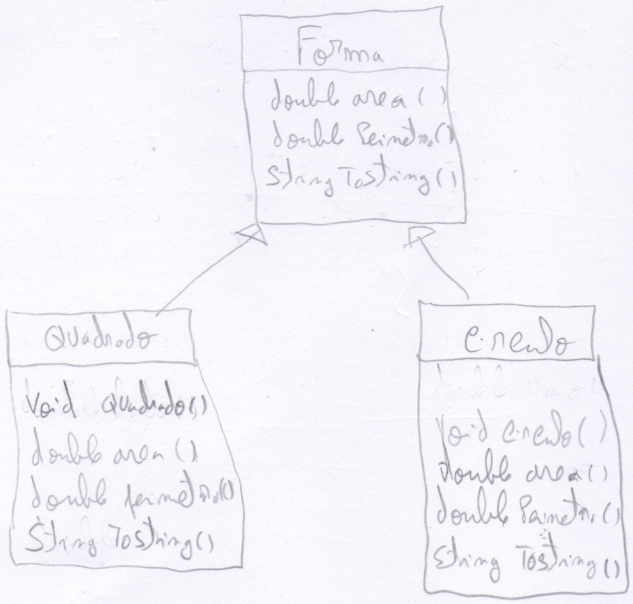
```
abstract class Forma {
    double area () { return 0; }
    double perimetro () { return 0; }
    String toString ();
};

class Quadrado extends Forma {
    double lado;
    void Quadrado (double lado) { this.lado = lado; }
    double area () { return lado * lado; }
    double perimetro () { return 4 * lado; }
    String toString () { return "Quadradro de lado " + lado; }
};

class Circulo extends Forma {
    double raio;
    void Circulo (double raio) { this.raio = raio; }
    double area () { return Math.PI * raio * raio; }
    double perimetro () { return 2 * Math.PI * raio; }
    String toString () { return "Circulo de raio " + raio; }
};
```

(a) (1v) Explicite a relação entre as 3 classes mencionadas, i.e. herança, características das classses, etc. Para isso, desenhe um diagrama com estas classes.



(b) (2v) Defina uma classe Retangulo com o posicionamento nesta hierarquia e comportamento esperados.
O construtor a definir deverá aceitar dois parâmetros do tipo double: um para a largura e outro para a altura. Inclua definições para os métodos area(), perimetro() e toString() apropriados para um retângulo.

```
Class Retangulo extends Forma {
    double lado1;
    double lado2;
Void Retangulo (double lado1, double lado2) {
    this.lado1 = lado1;
    this.lado2 = lado2;
double area () { return lado1 * lado2;}
double perimetro() { return 2 * (lado1 + lado2)}
String toString () { return "lado 1" + lado1 + "lado2" + lado2;}
}
```

(c) (1v) Se quisesse definir o Quadrado em termos do Retangulo, o que é que faria?

Bastava tirar a classe Retangulo que respondi na alínea (b)

(d) (1v) Considere que f é uma Coleção de formas, e que pretendemos saber qual delas a que tem a maior área. Para isso definimos um método maiorForma(). Este método poderá recorrer a todas mensagens que se podem enviar a objetos da classe Forma.
Sabe-se que a classe de f permite aceder a todas as formas que integram a coleção, recorrendo ao método **nextForma()**, que retorna cada uma das formas, em chamadas sucessivas.
Pretendemos que maiorForma() retorne a Forma do seu conjunto que tiver **maior área**, por exemplo, se tivermos:

```
1  Coleção f;
2  Forma maior;
3  f.acrescenta (new Quadrado (3.0)); // quadrado de lado 3
4  f.acrescenta (new Circulo (2.0));  // circulo de raio 2
5  f.acrescenta (new Retangulo (1.5, 2.5)); // retangulo de 1.5 x 2.5
6  maior = f.maiorForma ();
7  System.out.println ("A maior forma é " + maior);
```

Diga qual será o output deste troço de código. Assuma que as declarações de classe estão todas feitas e que o método toString() foi definido para as subclasses de Forma.

Area do quadrado = 3x3 = 9
Area do circulo = π x 2 x 2 = 12,566
Area do retangulo = (2 * 1.5) + (2 * 2.5) = 8
logo o output é a area do circulo
"A maior forma é " + maior
circulo (ou) 12,566

(e) (2v) Complete a definição da função maiorForma.

```
1   class Coleção {
2     ...;
3     Forma nextForma ();
4
5     Forma maiorForma () {
6       Forma aForma;
7       ...; // RESPONDA AQUI
8       return aForma;
9     }
10  }
```

Forma maiorForma()
Forma aForma;
maiorde2 = Math.max(Quadrado(3.0), Circulo(2.0));
maiorde3 = Math.max(maiorde2, Retangulo(1.5, 2.5));
if (maiorde3 == Quadrado(3.0)){
aForma = "Quadrado";
return aForma;
else if (maiorde3 == Circulo(2.0)){
aForma = "Circulo";
return aForma;
else if (maiorde3 == Retangulo(1.5, 2.5))
aForma = "Circulo";
return aForma;