Tema 1 – Sessió 6
Programació orientada a objecte

Exercici

Volem crear una jerarquia de classes que ens permeti representar i manipular figures geomètriques de tres tipus diferents: rectangle, triangle i cercle.

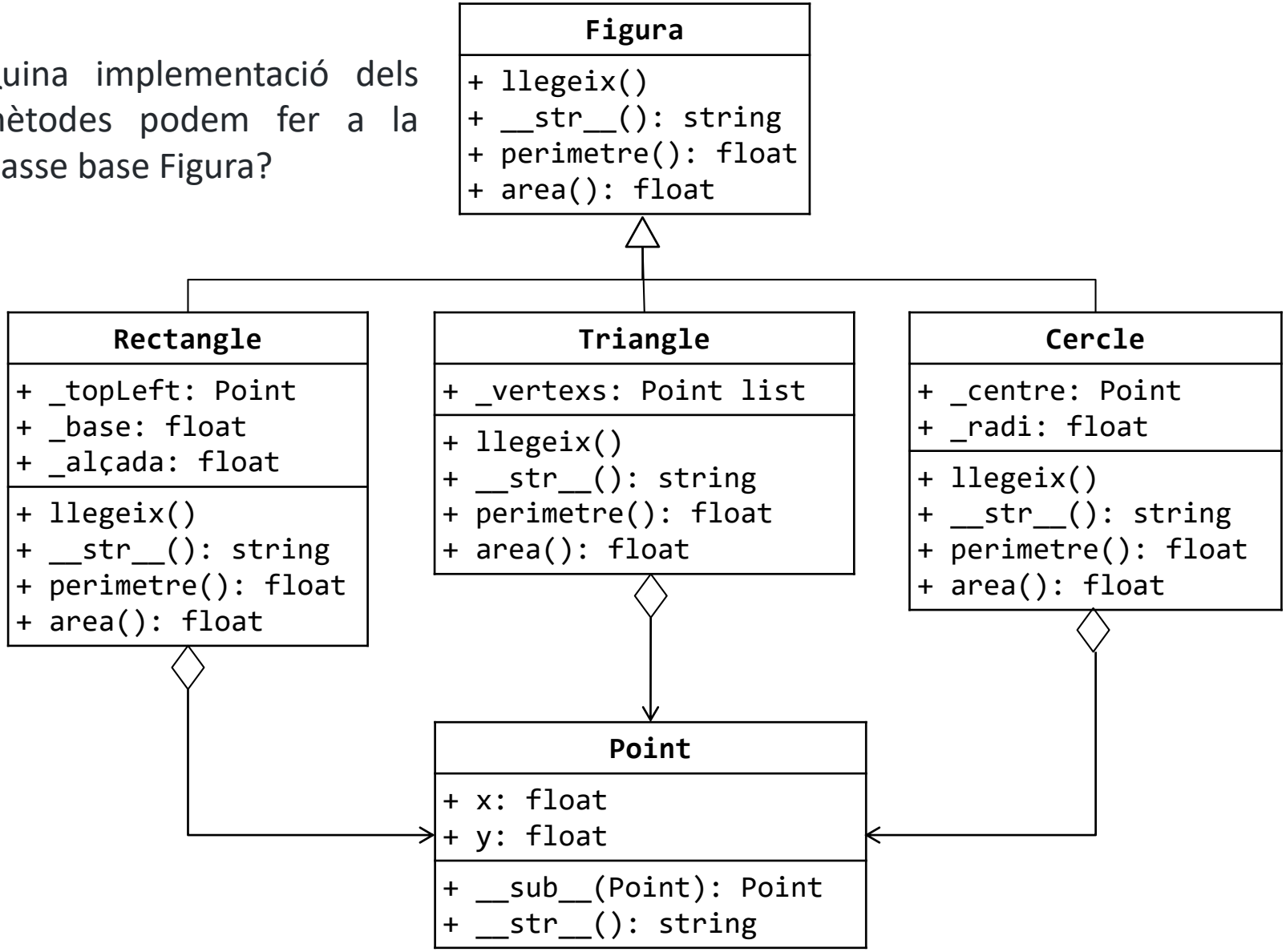
- Totes les figures geomètriques han de tenir mètodes que permetin calcular el perímetre, l'àrea, llegir les dades de la figura per teclat i mostrar-les per pantalla.
- Els rectangles es defineixen a partir de la cantonada superior esquerra i de la longitud de de la base i l'alçada.
- Els triangles es defineixen a partir de la posició de cadascun dels seus vèrtexs.
- Els cercles es defineixen a partir del punt del centre del cercle i de la longitud del radi.

Podem suposar que tenim la classe Point que hem utilitzat en exercicis anteriors

Point
+ x: float + y: float
+ __sub__(Point): Point + __str__(): string

Classes abstractes

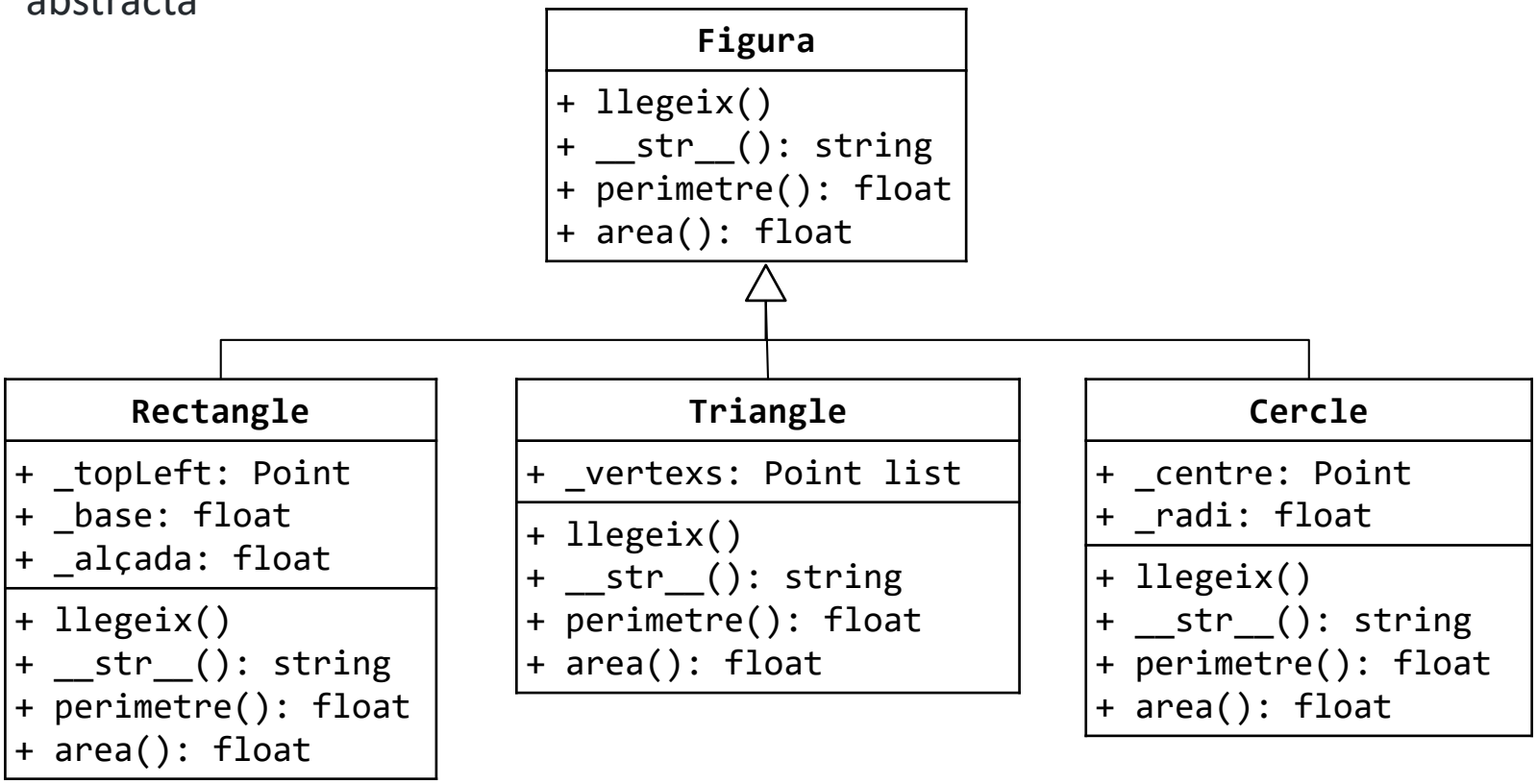
Quina implementació dels mètodes podem fer a la classe base Figura?



Classes abstractes

La classe `Figura` és una **classe abstracta**:

- No proporciona una implementació concreta de tots els mètodes que defineix
- Només proporciona la definició (nom i paràmetres) dels mètodes
- La implementació dels mètodes s'ha de fer obligatòriament a les subclasses.
- La classe base `Figura` només defineix una interfície comuna per totes les seves classes derivades. Normalment no tindrà sentit crear objectes d'una classe base abstracta



Classes abstractes

```
import abc

class Figura(metaclass=abc.ABCMeta):
    @abc.abstractmethod
    def area(self):
        raise NotImplementedError()

    @abc.abstractmethod
    def perimetre(self):
        raise NotImplementedError()

    @abc.abstractmethod
    def llegeix(self):
        raise NotImplementedError()

    @abc.abstractmethod
    def __str__(self):
        raise NotImplementedError()
```

→ Indica que la classe és abstracta

Mètode abstracte:

- No implementat a la classe base
- L'han d'implementar les classes derivades