Object-Oriented Programming: Application to the Java Language

Abstract classes

Abstract classes: benefits

- We don't always know the default behavior of an operation that is common to several subclasses
  - Example: the roof of a *voiture décapotable*. We know that all *décapotables* can fold their roof, but the mechanism differs from one *décapotable* to another
  - Solution: we can declare the method as abstract in the parent class and not provide a default implementation
- Abstract method and consequences: 3 rules to remember
  - If even one method in a class is abstract, then the class itself becomes abstract
  - You cannot instantiate an abstract class because at least one of its methods has no implementation
  - All child classes inheriting from the abstract parent class must implement all of its abstract methods, otherwise they are also abstract

Abstract classes and Java

- The keyword **abstract** is used to specify that a class is abstract
- An abstract class is declared like this:

```
public abstract class NomMaClasse {
   ...
}
```

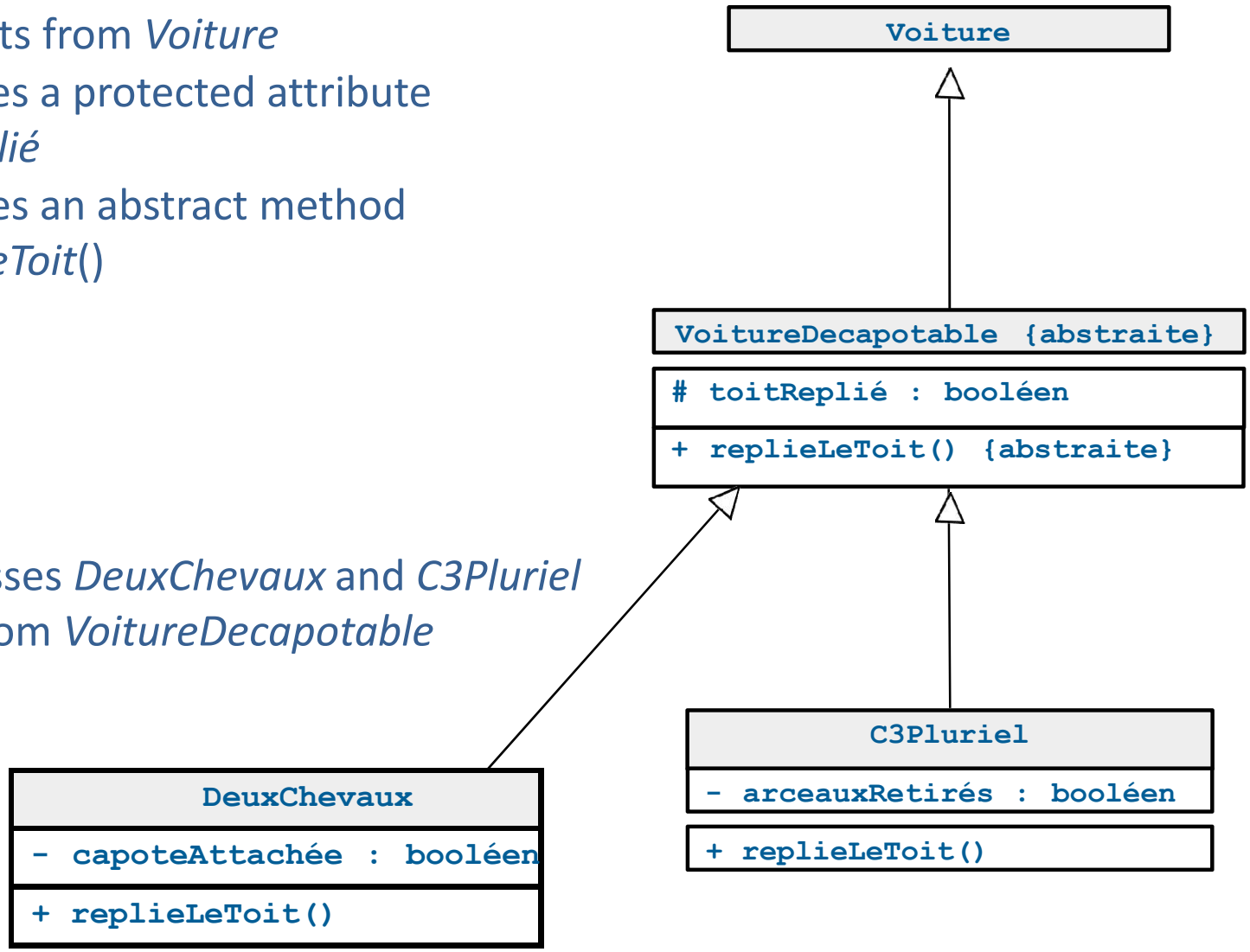
- An abstract method is declared like this:

```
public abstract void maMethode(...);
```

To create an abstract method, the signature (name and parameters) is declared without specifying the body, and the abstract keyword is added

Abstract classes: example VoitureDecapotable

- The class *VoitureDecapotable*
  - Inherits from *Voiture*
  - Defines a protected attribute *toitReplié*
  - Defines an abstract method *replieLeToit*()



- The classes *DeuxChevaux* and *C3Pluriel* inherit from *VoitureDecapotable*

Abstract classes: example VoitureDecapotable

- Example: a voiture décapotable

Abstract class

Abstract method

```
public abstract class VoitureDecapotable
    extends Voiture {
    protected boolean toitReplié;

    public abstract void replieLeToit();
}
```

```
public class DeuxChevaux extends VoitureDecapotable {
    private boolean capoteAttachee;

    public void replieLeToit() {
        this.toitReplie = true;
        this.capoteAttachee = true;
    }
}
```

```
public class C3Pluriel extends VoitureDecapotable {
    private boolean arceauxRetirés;

    public void replieLeToit() {
        this.toitReplie = true;
        this.arceauxRetirés = true;
    }
}
```

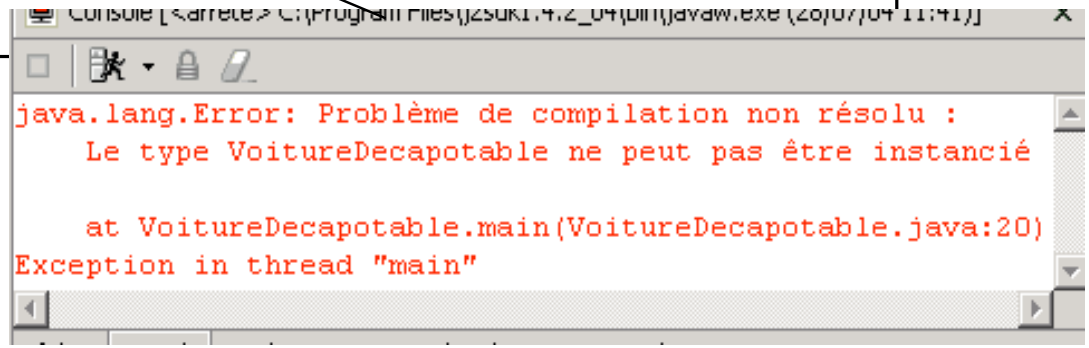
Attention: this is not overriding. We are talking about implementing an abstract method

Abstract classes: example VoitureDecapotable

- Example (continued): a voiture décapotable

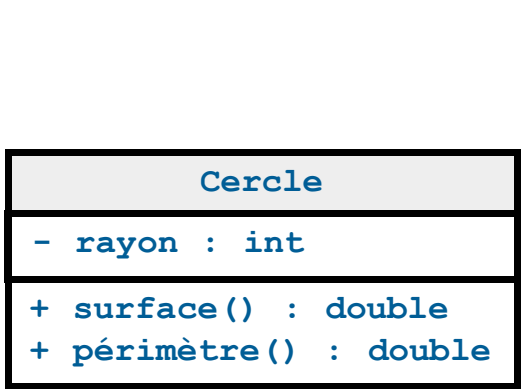
```
public class Test {
    public static void main (String[] argv) {
        // Déclaration et création d'une DeuxCheveaux
        VoitureDecapotable voitureAncienne = new DeuxCheveaux(...);
        // Envoi de message
        voitureAncienne.replieLeToit();

        // Déclaration et création d'une C3Pluriel
         VoitureDecapotable voitureRecente = new C3Pluriel(...);
        // Envoi de message
        voitureRecente.replieLeToit();

        // Déclaration et création d'une VoitureDecapotable
        VoitureDecapotable voitureDecapotable =
            new VoitureDecapotable(...); //
            Erreur
    }
}
```

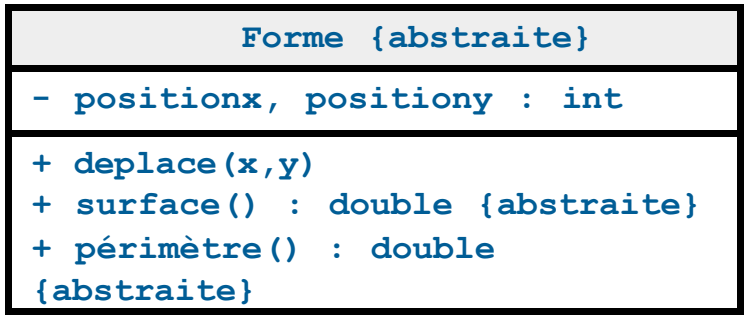
Attention: The class *VoitureDecapotable* cannot be instantiated because it is abstract



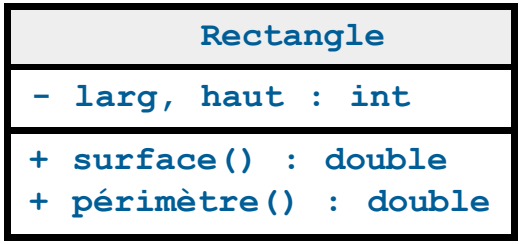
Abstract classes: example Forme

- Example: the class *Forme*
  - The methods *surface*() and *périmètre*() are abstract
  - These methods only make "sense" for the subclasses *Cercle* and *Rectangle*






```
public abstract class Forme {
    private int positionx, positiony;

    public void deplacer(double dx, double dy){
        x += dx; y += dy;
    }

    public abstract double périmètre();
    public abstract double surface();
}
```



No implementation!!