Scope and extend of data

COMP 102

Victoria University of Wellington

Places: variables *vs* fields

- Two kinds of places to store information:
- Variables (including parameters)
  - defined inside a method
  - temporary – information is lost when worksheet is finished
  - new place created *every time method is called* (each worksheet)
  - only accessible from inside the method.

- Fields
  - defined inside a class, but not inside a method
  - long term – information lasts as long as the object
  - new place created *for each object*
  - accessible from all methods in the class, and from constructor.

Extent and scope

- A place with a value must be accessible to some code at some time.
- **Extent**: how long it will be accessible
  - local variables (and parameters) in methods have a limited extent
    ⇒ only until the end of the current invocation of the method
  - fields have indefinite extent
    ⇒ as long as the object exists
- **Scope**: what parts of the code can access it
  - Full scope rules are more complicated
    - local variables: accessible only to statements
      - after the declaration
      - inside the block { … } containing the declaration
    - fields: at least visible to the containing class; maybe further.

Scope of variables

```
//read info from file and display
while (scan.hasNext() ){
      String ans = scan.next();
      if ( ans.equals("flower") ) {
            Color  center = Color.red;
            int diam = 30;
      }
      else if (ans.equals("bud") ) {
            Color  center = Color.green;
            int diam = 15;
      }
        :
      UI.setColor(center);
      UI.fillOval(x, y, diam, diam);
        :
}
```

different variables!

Out of scope

```
while (scan.hasNext() ){
      String ans = scan.next();
      Color center;
      int diam;
      if ( ans.equals("flower") ) {
            center = Color.red;
            diam = 15;
      }
      else if (ans.equals("bud") ) {
            center = Color.blue;
            diam = 30;
      }
        :
      UI.setColor(center);
      UI.fillOval(x, y, diam, diam);
        :
}
```

may not be intialised

How do you fix it?

Fields: scope, visibility, encapsulation

- Fields are accessible to all code in all the (ordinary) methods in the class.
- Should they be accessible to methods in other classes?
  - ⇒ **visibility**: **public** or **private**
  - **public** means that methods in other classes can access the fields

    cfg1.figX = 30 in the CartoonStory class would be OK
  - **private** means that methods in other classes **cannot** access the fields

    cfg1.figX = 30 in the CartoonStory class would be an error.

The principle of encapsulation says

- Keep fields private.
- Provide methods to access and modify the fields, if necessary