CS133: Developing Programming Principles

Lecture 6

Parameters, Return Types, Information Hiding, Assertions, Accessor/Mutator Methods

Parameters

- A parameter is information that a method must receive from the user.
- Since the method doesn't know the value, it treats it as a variable within the method.
- Name and type are identified in the method's signature.

Parameters continued

- Examples:

  `String` class

```
public char charAt(int position)
```

  `CashRegister` class

```
public void addToBalance(
       double amount)
```

- Position and amount are parameters.

Parameters vs. Arguments

- A parameter is the variable the method declares and manipulates inside the method.
- An argument is a specified value for the parameter when the method is invoked.

The return statement

- Methods that return a value must include a statement.

  return value;
- Where value is of ***ReturnType***.
- Example: CashRegister class

```
public double getBalance()
{
   return this.balance;
}
```


Information hiding

- The act of designing methods in a way that doesn't require the user to understand **how** they work, only **what they do**.
- Why hide information?
  - Simplicity
  - Security
  - Stability

Precondition

- A comment that lists all conditions that must be satisfied for a method to execute correctly.
- Placed above the method signature.

Postcondition

- A comment that explains all the possible outputs of the method.
- Placed immediately after the precondition.

Example

```
/* Pre:                                  */
/* Post:                                 */
public int exponent(int base,int exp)
{


}
```


Pre and postconditions

- Together, the two comments allow a user to know exactly how to call the method and what output to expect.
- User can use method without knowing how it works.
- User can write code using the method before it exists.

Example

String class

```
public char charAt(int position)
/* Pre:                 */
/* Post:                */
```

Board class

```
public void putPeg (Color colour,
          int position)
/* Pre:                 */
/* Post:                */
```


Assertions

- Comments that assert a certain boolean statement is true at a given point.
- Very useful in testing.
- For assertion checks, refer to p. 252-254 of the text.
- Used in CS134.

Public vs. private modifiers

- **`private`** – only accessible within the class. More secure.
- **`public`** – available both inside and outside the class.

`Board` class

- These are the signatures of some **`Board`** methods:

```
public void putPeg(Color colour,
                   int position)
public int getPosition()
private Color invisible(int row,
                        int col)
```

Instance variables

- Should **always** be private.
- But what if they need to be accessed from outside the class?

Board template

```
class Board {
  // Instance variables
  private Color[][] grid;
  private Coordinate lastClick;
  private String message = "";
  private int[] line = new int[4];
  private boolean drawLine = false;
  // Methods…
}
```

Accessor and mutator methods

- **Accessor method** – returns the value of a private variable.
- **Mutator method** – modifies the value of a private variable.

Not every instance variable needs accessor or mutator methods!

Accessor example

```
private int theLength = 2;

public int getLength()
{
   return this.theLength;
}
```

Accessors in **`Board`** class

- There are two accessor methods in the **`Board`** class:

```
public int getRows();
public int getColumns();
```


Mutator example

```
private int theLength = 2;

public void setLength(int newLength)
{
   this.theLength = newLength;
}
```

Mutators in **`Board`** class

- There are six mutator methods in the **`Board`** class:

```
public void putPeg(Color colour, int col)
public void putPeg(Color colour, int row,
                   int col)
public void removePeg(int col)
public void removePeg(int row, int col)
public void drawLine(int x1, int y1,
                     int x2, int y2)
public void displayMessage(
                         String theMessage)
```


Why do this?

- **Accessor** – can check if the user is permitted to access the value before disclosing it.
- **Mutator** – before changing the value of the instance variable, it can check if the new value is acceptable.

Summary

- Parameters
- Information hiding
- Preconditions and postconditions
- Assertions
- Public vs. private modifiers
- Accessor and mutator methods

ERROR: undefined
OFFENDING COMMAND:

STACK: