Substitution Principle

public class Tree {
    public Tree getChild (int n)
    // REQUIRES: 0 <= n < children.length
    // EFFECTS: Returns the Tree that is the nth leftmost child
    // of this. NOTE: the rep is exposed!
}

public class BinaryTree extends Tree {
    // OVERVIEW: A BinaryTree is a mutable tree where the nodes are
    // int values and each node has zero, one or two children.
    @Override
    public BinaryTree getChild (int n)
    // REQUIRES: 0 <= n < 2
    // EFFECTS: If this has at least n children, returns a copy of the
    // BinaryTree that is the nth leftmost child of this. Otherwise,
    // returns null.
}

Does \text{pre}_{\text{Tree}} \text{ imply } \text{pre}_{\text{BinaryTree}} ?

Substitution Principle

public class Tree {
    public Tree getChild (int n)
    // REQUIRES: 0 <= n < children.length
    // EFFECTS: Returns the Tree that is the nth leftmost child
    // of this. NOTE: the rep is exposed!
}

public class BinaryTree extends Tree {
    // OVERVIEW: A BinaryTree is a mutable tree where the nodes are
    // int values and each node has zero, one or two children.
    @Override
    public BinaryTree getChild (int n)
    // REQUIRES: 0 <= n < 2
    // EFFECTS: If this has at least n children, returns a copy of the
    // BinaryTree that is the nth leftmost child of this. Otherwise,
    // returns null.

    Does \text{post}_{\text{BinaryTree}} \text{ imply } \text{post}_{\text{Tree}} ?

Code that “breaks” with subtype:

```java
Tree b1 = t.getChild(0);
Tree b2 = t.getChild(0);
assert b1 == b2;
```
Parameterized Filters

Option 1:
Filter subtype
- PointFilter
- NegativeFilter
- BlurFilter
- MultiFilter
- AverageFilter
- ParameterizedFilter

Option 2:
- public int getParameter();
- public abstract boolean hasParameter();

Option 3:
- public class BlurFilter extends Filter implements ParameterizedFilter {
  private int param;
  public void setParameter(int val) {
    param = val;
  }
  public int getParameter() {
    return param;
  }
  public String getPrompt() {
    return "Enter the blurring factor: ";
  }
  ...
}

- public interface ParameterizedFilter {
  public void setParameter(int val);
  public int getParameter();
  public String getPrompt();
}