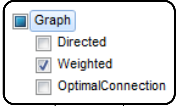
Why is preprocessor wilderness a problem?

```
class Edge {
  Node first, second;
//#ifdef Weighted
  int weight;
//#endif
  Edge(Node first, Node second) {...}
  boolean equals(Edge e) {
    return (first == e.first
      && second == e.first
//#ifndef Directed
      || first == e.second
      && second == e.first
      ) && weight == e.weight;
//#endif
  }
  void testEquality() {
    Node a = new Node();
    Node b = new Node();
    Edge e = new Edge(a, b);
    Assert.assertTrue(e.equals(e));
} }
```

Why is preprocessor wilderness a problem?

```
class Edge {
  Node first, second;
//#ifdef Weighted
  int weight;
//#endif
  Edge(Node first, Node second) {...}
  boolean equals(Edge e) {
    return (first == e.first
      && second == e.first
//#ifndef Directed
      || first == e.second
      && second == e.first
      ) && weight == e.weight;
//#endif
  }
  void testEquality() {
    Node a = new Node();
    Node b = new Node();
    Edge e = new Edge(a, b);
    Assert.assertTrue(e.equals(e));
} }
```

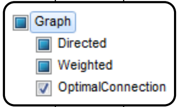


- Graph
  - [ ] Directed
  - [✓] Weighted
  - [ ] OptimalConnection

```
class Edge {
  Node first, second;

  int weight;

  Edge(Node first, Node second) {...}
  boolean equals(Edge e) {
    return (first == e.first
      && second == e.first

      || first == e.second
      && second == e.first
      ) && weight == e.weight;

  }
  void testEquality() {
    Node a = new Node();
    Node b = new Node();
    Edge e = new Edge(a, b);
    Assert.assertTrue(e.equals(e));
} }
```

Why is preprocessor wilderness a problem?

```
class Edge {
   Node first, second;
//#ifdef Weighted
   int weight;
//#endif
   Edge(Node first, Node second) {...}
   boolean equals(Edge e) {
      return (first == e.first
         && second == e.first
//#ifndef Directed
         || first == e.second
         && second == e.first
         ) && weight == e.weight;
//#endif
   }
   void testEquality() {
      Node a = new Node();
      Node b = new Node();
      Edge e = new Edge(a, b);
      Assert.assertTrue(e.equals(e));
} }
```

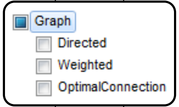


- ▣ Graph
  - ▣ Directed
  - ▣ Weighted
  - ✓ OptimalConnection

```
class Edge {
   Node first, second;

   int weight;

   Edge(Node first, Node second) {...}
   boolean equals(Edge e) {
      return (first == e.first
         && second == e.first



   }
   void testEquality() {
      Node a = new Node();
      Node b = new Node();
      Edge e = new Edge(a, b);
      Assert.assertTrue(e.equals(e));
} }
```

Why is preprocessor wilderness a problem?

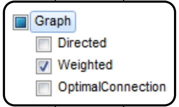
```
class Edge {
  Node first, second;
//#ifdef Weighted
  int weight;
//#endif
  Edge(Node first, Node second) {...}
  boolean equals(Edge e) {
    return (first == e.first
      && second == e.first
//#ifndef Directed
      || first == e.second
      && second == e.first
      ) && weight == e.weight;
//#endif
  }
  void testEquality() {
    Node a = new Node();
    Node b = new Node();
    Edge e = new Edge(a, b);
    Assert.assertTrue(e.equals(e));
} }
```



```
class Edge {
  Node first, second;



  Edge(Node first, Node second) {...}
  boolean equals(Edge e) {
    return (first == e.first
      && second == e.first

      || first == e.second
      && second == e.first
      ) && weight == e.weight  ;

  }
  void testEquality() {
    Node a = new Node();
    Node b = new Node();
    Edge e = new Edge(a, b);
    Assert.assertTrue(e.equals(e));
} }
```

Why is preprocessor wilderness a problem?

```
class Edge {
  Node first, second;
//#ifdef Weighted
  int weight;
//#endif
  Edge(Node first, Node second) {...}
  boolean equals(Edge e) {
    return (first == e.first
      && second == e.first
//#ifndef Directed
      || first == e.second
      && second == e.first
      ) && weight == e.weight;
//#endif
  }
  void testEquality() {
    Node a = new Node();
    Node b = new Node();
    Edge e = new Edge(a, b);
    Assert.assertTrue(e.equals(e));
} }
```



- Graph
  - [ ] Directed
  - [✓] Weighted
  - [ ] OptimalConnection

```
class Edge {
  Node first, second;

  int weight;

  Edge(Node first, Node second) {...}
  boolean equals(Edge e) {
    return (first == e.first
      && second == e.first

      || first == e.second
      && second == e.first
      ) && weight == e.weight;

  }
  void testEquality() {
    Node a = new Node();
    Node b = new Node();
    Edge e = new Edge(a, b);
    Assert.assertTrue(e.equals(e));
} }
```