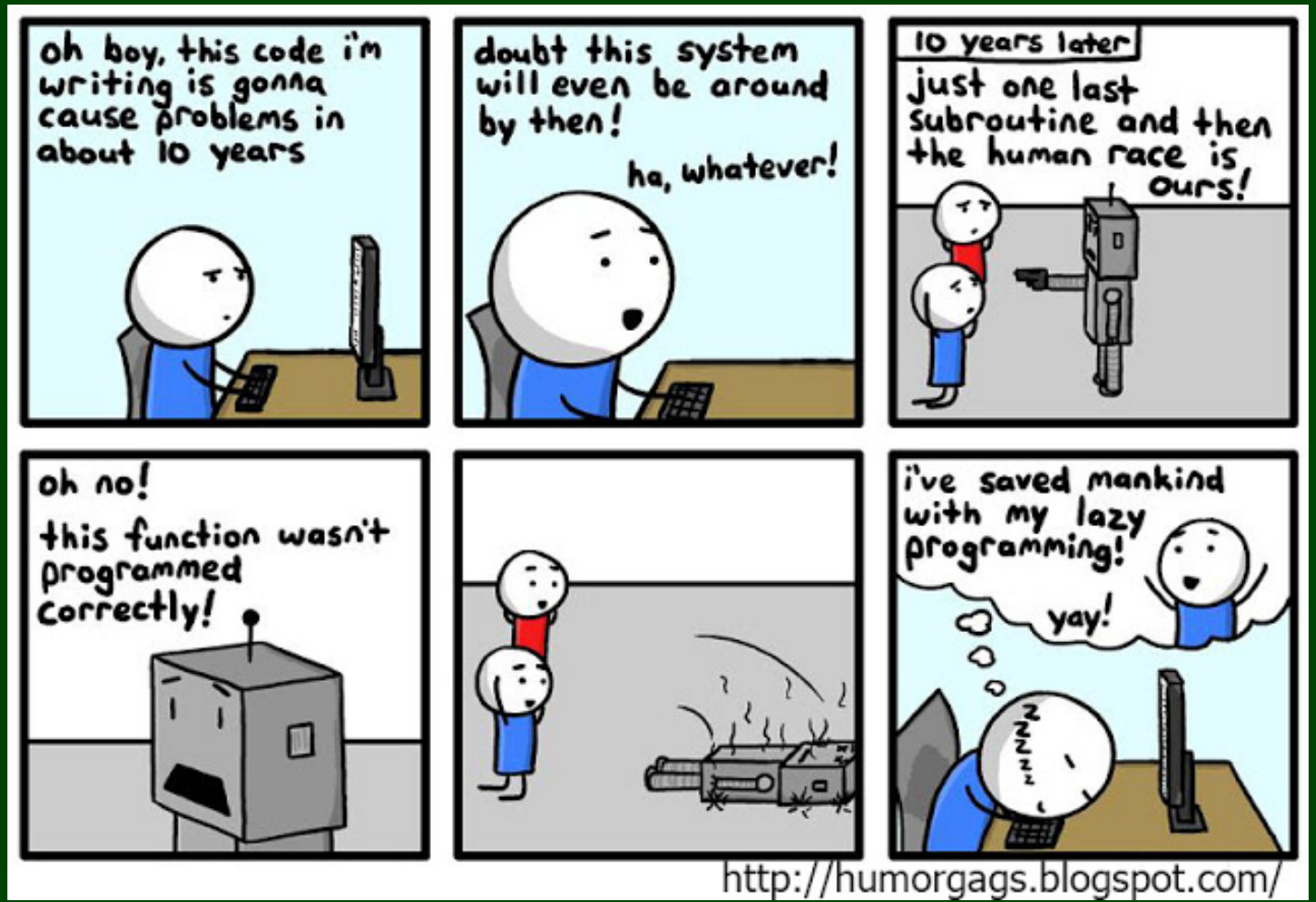
CSE 143

Computer Programming II

More BSTs


oh boy, this code i'm writing is gonna cause problems in about 10 years
doubt this system will even be around by then!
ha, whatever!
10 years later
just one last subroutine and then the human race is ours!
oh no!
this function wasn't programmed correctly!
i've saved mankind with my lazy programming!
yay!
zzzzz
http://humorgags.blogspot.com/

Outline

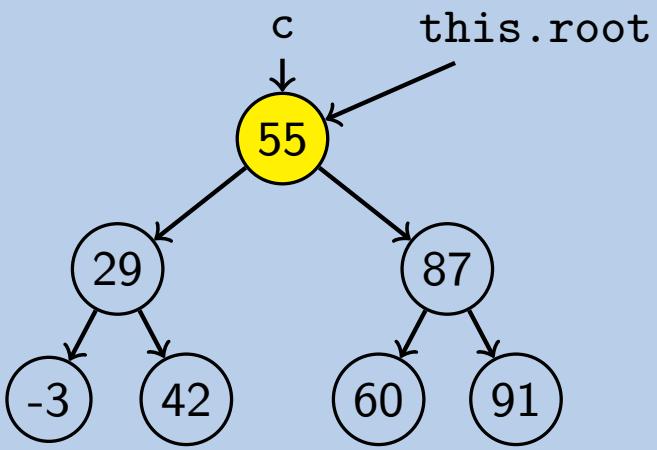
Adding to a BST

Code

```
1  private IntTreeNode add(IntTreeNode c, int value) {
2     if (c == null) {
3        c = new IntTreeNode(value);
4     }
5     else if (c.data > value) { // 55 > 49
6        c.left = add(c.left, value);
7     }
8     else if (c.data < value) {
9        c.right = add(c.right, value);
10    }
11    return c;
12 }
```

Example (`tree.add(49)`)

value = 49



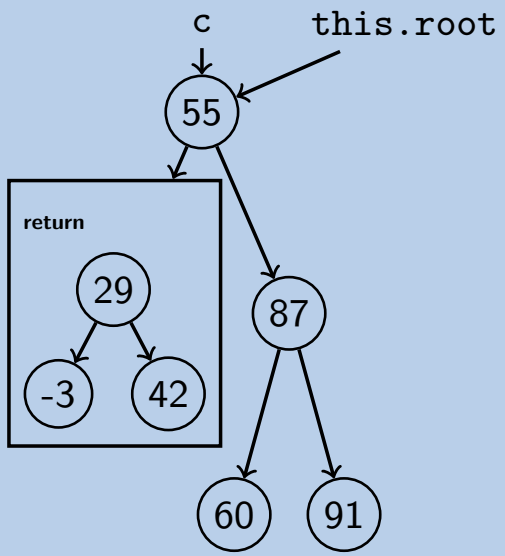
Adding to a BST

Code

```
1  private IntTreeNode add(IntTreeNode c, int value) {
2     if (c == null) {
3        c = new IntTreeNode(value);
4     }
5     else if (c.data > value) {
6        c.left = add(c.left, value);
7     }
8     else if (c.data < value) {
9        c.right = add(c.right, value);
10    }
11    return c;
12 }
```

Example (`tree.add(49)`)

`value = 49`



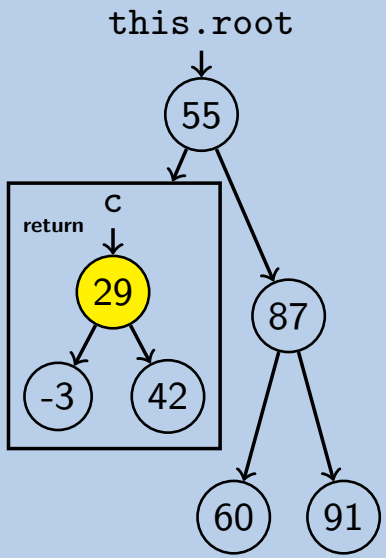
Adding to a BST

Code

```
1  private IntTreeNode add(IntTreeNode c, int value) {
2     if (c == null) {
3        c = new IntTreeNode(value);
4     }
5     else if (c.data > value) {
6        c.left = add(c.left, value);
7     }
8     else if (c.data < value) { // 29 < 49
9        c.right = add(c.right, value);
10    }
11    return c;
12 }
```

Example (`tree.add(49)`)

`value = 49`



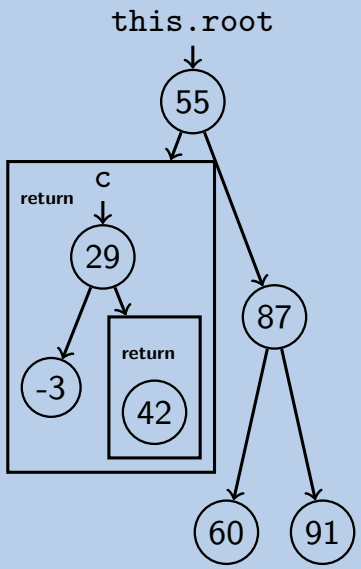
Adding to a BST

Code

```
1  private IntTreeNode add(IntTreeNode c, int value) {
2     if (c == null) {
3        c = new IntTreeNode(value);
4     }
5     else if (c.data > value) {
6        c.left = add(c.left, value);
7     }
8     else if (c.data < value) { // 29 < 49
9        c.right = add(c.right, value);
10    }
11    return c;
12 }
```

Example (`tree.add(49)`)

`value = 49`



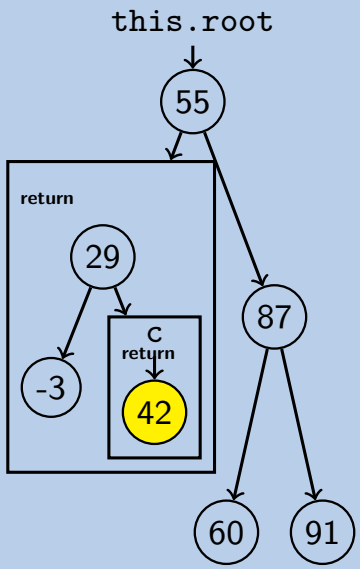
Adding to a BST

Code

```
1  private IntTreeNode add(IntTreeNode c, int value) {
2     if (c == null) {
3        c = new IntTreeNode(value);
4     }
5     else if (c.data > value) {
6        c.left = add(c.left, value);
7     }
8     else if (c.data < value) { // 42 < 49
9        c.right = add(c.right, value);
10    }
11    return c;
12 }
```

Example (`tree.add(49)`)

`value = 49`



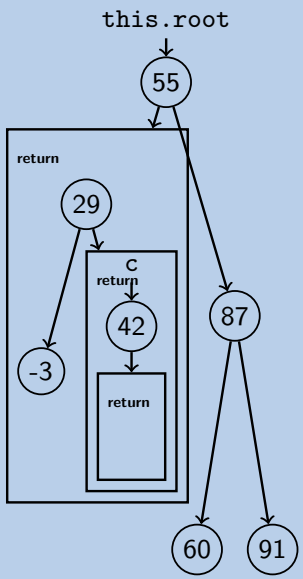
Adding to a BST

Code

```
1  private IntTreeNode add(IntTreeNode c, int value) {
2     if (c == null) {
3        c = new IntTreeNode(value);
4     }
5     else if (c.data > value) {
6        c.left = add(c.left, value);
7     }
8     else if (c.data < value) { // 42 < 49
9        c.right = add(c.right, value);
10    }
11    return c;
12 }
```

Example (`tree.add(49)`)

`value = 49`



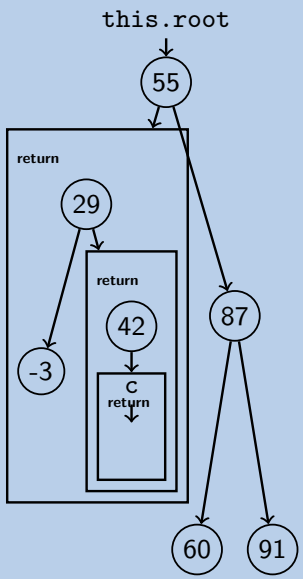
Adding to a BST

Code

```
1  private IntTreeNode add(IntTreeNode c, int value) {
2     if (c == null) {
3        c = new IntTreeNode(value);
4     }
5     else if (c.data > value) {
6        c.left = add(c.left, value);
7     }
8     else if (c.data < value) {
9        c.right = add(c.right, value);
10    }
11    return c;
12 }
```

Example (`tree.add(49)`)

`value = 49`



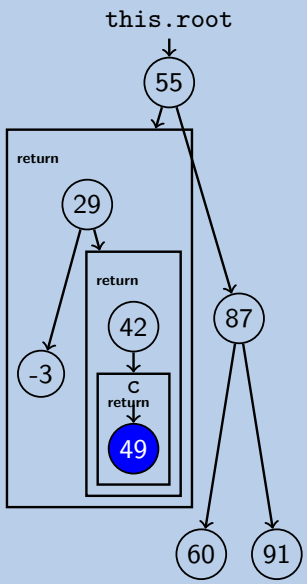
Adding to a BST

Code

```
1  private IntTreeNode add(IntTreeNode c, int value) {
2     if (c == null) {
3        c = new IntTreeNode(value);
4     }
5     else if (c.data > value) {
6        c.left = add(c.left, value);
7     }
8     else if (c.data < value) {
9        c.right = add(c.right, value);
10    }
11    return c;
12 }
```

Example (`tree.add(49)`)

`value = 49`



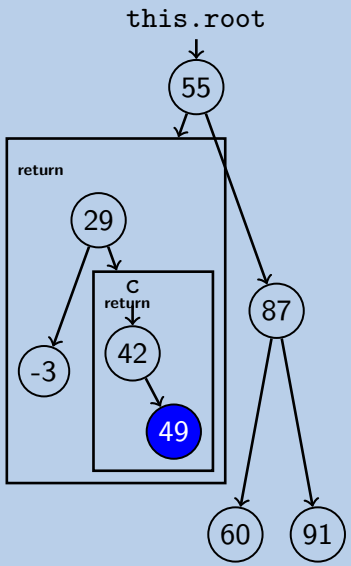
Adding to a BST

Code

```
1  private IntTreeNode add(IntTreeNode c, int value) {
2     if (c == null) {
3        c = new IntTreeNode(value);
4     }
5     else if (c.data > value) {
6        c.left = add(c.left, value);
7     }
8     else if (c.data < value) {
9        c.right = add(c.right, value);
10    }
11    return c;
12 }
```

Example (`tree.add(49)`)

`value = 49`



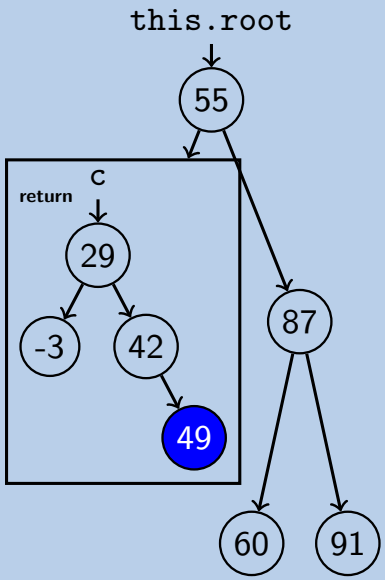
Adding to a BST

Code

```
1  private IntTreeNode add(IntTreeNode c, int value) {
2     if (c == null) {
3        c = new IntTreeNode(value);
4     }
5     else if (c.data > value) {
6        c.left = add(c.left, value);
7     }
8     else if (c.data < value) {
9        c.right = add(c.right, value);
10    }
11    return c;
12 }
```

Example (`tree.add(49)`)

`value = 49`



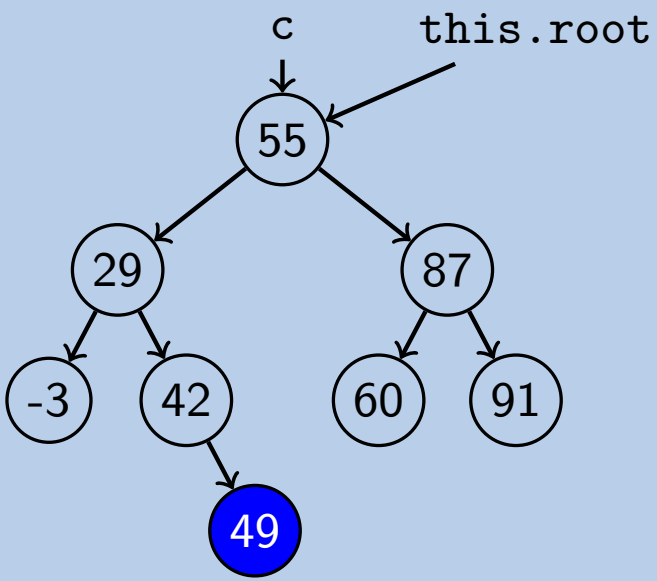
Adding to a BST

Code

```
1  private IntTreeNode add(IntTreeNode c, int value) {
2     if (c == null) {
3        c = new IntTreeNode(value);
4     }
5     else if (c.data > value) {
6        c.left = add(c.left, value);
7     }
8     else if (c.data < value) {
9        c.right = add(c.right, value);
10    }
11    return c;
12 }
```

Example (`tree.add(49)`)

`value = 49`



first

Write a function `first` in the BST class with the following signature:

```
public int first();
```

that returns the smallest value in the tree. If the tree is empty, `first` should throw a `NoSuchElementException`.

```
1  public int first() {
2     return first(this.root);
3  }
4
5  private int first(IntTreeNode current) {
6     if (current == null) {
7        throw new NoSuchElementException();
8     }
9     /* Keep on going left as far as we can */
10    else if (current.left != null) {
11       return first(current.left);
12    }
13    else {
14       return current.data;
15    }
16 }
```

`remove` Solution

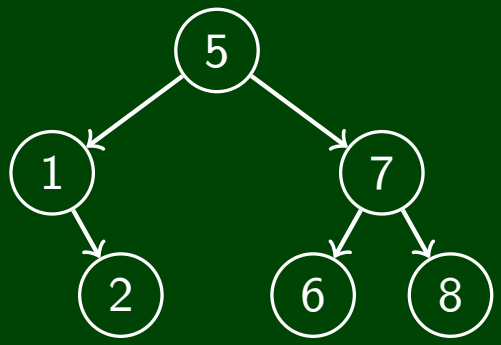
`remove`

Write a function `remove` in the BST class with the following signature:

```
public void remove(int value);
```

that removes `value` from the tree if it exists.

Consider the following tree:



Let's try the following removals:

- `tree.remove(2)`
- `tree.remove(1)`
- `tree.remove(7)`
- `tree.remove(5)`

Removals

`tree.remove(2)`

`tree.remove(1)`

`tree.remove(7)`

`tree.remove(5)`

Removals

`tree.remove(2)`

`tree.remove(1)`

`tree.remove(7)`

`tree.remove(5)`

`remove` Solution

```
 1 public void remove(int value) {
 2    this.root = remove(this.root, value);
 3 }
 4 private IntTreeNode remove(IntTreeNode current, int value) {
 5    if (current == null) { return null; }
 6    else if (current.data > value) {
 7       current.left = remove(current.left, value);
 8    }
 9    else if (current.data < value) {
10       current.right = remove(current.right, value);
11    }
12    else if (current.left == null && current.right == null) {
13       current = null;
14    }
15    else if (current.left == null && current.right != null) {
16       current = current.right;
17    }
18    else if (current.left != null && current.right == null) {
19       current = current.left;
20    }
21    else {
22       current.data = first(current.right);
23       current.right = remove(current.right, current.data);
24    }
25    return current;
26 }
```