State traces test

The INFDEV team

Hogeschool Rotterdam
Rotterdam, Netherlands

```
1  class MyClass {
2    static public int f(int x) {
3      return (x + 10);
4    }
5  }
6  ...
7  Console.WriteLine(MyClass.f(10))
```

Stack:

PC
1

```
1  class MyClass {
2    static public int f(int x) {
3      return (x + 10);
4    }
5  }
6  ...
7  Console.WriteLine(MyClass.f(10))
```

Stack:

PC
7

```
1  class MyClass {
2    static public int f(int x) {
3      return (x + 10);
4    }
5  }
6  ...
7  Console.WriteLine(MyClass.f(10))
```

Stack:

PC	...		PC	ret	x
7	...		3	null	10

```
1  class MyClass {
2    static public int f(int x) {
3      return (x + 10);
4    }
5  }
6  ...
7  Console.WriteLine(MyClass.f(10))
```

Stack:

PC	...		PC	ret
7	...		3	20

```
1  class MyClass {
2    static public int f(int x) {
3      return (x + 10);
4    }
5  }
6  ...
7  Console.WriteLine(MyClass.f(10))
```

Stack:

PC
8

Output: 20

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 ...
11 Counter c = new Counter();
12 c.incr(5);
```

Stack:

PC
1

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 ...
11 Counter c = new Counter();
12 c.incr(5);
```

Stack:

PC
11

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 ...
11 Counter c = new Counter();
12 c.incr(5);
```

Stack:

PC
11

Heap:

1
cnt=

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 ...
11 Counter c = new Counter();
12 c.incr(5);
```

Stack:

PC	...		PC	ret	this
11	...		4	null	ref 1

Heap:

1
cnt=

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 ...
11 Counter c = new Counter();
12 c.incr(5);
```

Stack:

PC	...		PC	ret
11	...		4	null

Heap:

1
cnt=0

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 ...
11 Counter c = new Counter();
12 c.incr(5);
```

Stack:

PC	c
12	ref 1

Heap:

1
cnt=0

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 ...
11 Counter c = new Counter();
12 c.incr(5);
```

Stack:

PC	...		PC	ret	diff	this
12	...		7	null	5	ref 1

Heap:

1
cnt=0

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 ...
11 Counter c = new Counter();
12 c.incr(5);
```

Stack:

PC	...		PC	ret
12	...		7	null

Heap:

1
cnt=5

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 ...
11 Counter c = new Counter();
12 c.incr(5);
```

Stack:

PC	c
13	ref 1

Heap:

1
cnt=5

```
1  interface ICounter {
2    void Incr(int diff);
3  }
4  class Counter : ICounter {
5    private int cnt;
6    public Counter() {
7      this.cnt = 0;
8    }
9    public void Incr(int diff) {
10     this.cnt = (this.cnt + diff);
11   }
12 }
13 ICounter c = new Counter();
14 c.Incr(5);
```

Stack:

PC
1

```
1  interface ICounter {
2    void Incr(int diff);
3  }
4  class Counter : ICounter {
5    private int cnt;
6    public Counter() {
7      this.cnt = 0;
8    }
9    public void Incr(int diff) {
10     this.cnt = (this.cnt + diff);
11   }
12 }
13 ICounter c = new Counter();
14 c.Incr(5);
```

Stack:

PC
13

Heap:

1
cnt=

```
1  interface ICounter {
2    void Incr(int diff);
3  }
4  class Counter : ICounter {
5    private int cnt;
6    public Counter() {
7      this.cnt = 0;
8    }
9    public void Incr(int diff) {
10     this.cnt = (this.cnt + diff);
11   }
12 }
13 ICounter c = new Counter();
14 c.Incr(5);
```

Stack:

PC	...		PC	ret	this
13	...		7	null	ref 1

Heap:

1
cnt=

```
1  interface ICounter {
2    void Incr(int diff);
3  }
4  class Counter : ICounter {
5    private int cnt;
6    public Counter() {
7      this.cnt = 0;
8    }
9    public void Incr(int diff) {
10     this.cnt = (this.cnt + diff);
11   }
12 }
13 ICounter c = new Counter();
14 c.Incr(5);
```

Stack:

PC	...		PC	ret
13	...		7	null

Heap:

1
cnt=0

```
1  interface ICounter {
2    void Incr(int diff);
3  }
4  class Counter : ICounter {
5    private int cnt;
6    public Counter() {
7      this.cnt = 0;
8    }
9    public void Incr(int diff) {
10     this.cnt = (this.cnt + diff);
11   }
12 }
13 ICounter c = new Counter();
14 c.Incr(5);
```

Stack:

PC	c
14	ref 1

Heap:

1
cnt=0

```
1  interface ICounter {
2    void Incr(int diff);
3  }
4  class Counter : ICounter {
5    private int cnt;
6    public Counter() {
7      this.cnt = 0;
8    }
9    public void Incr(int diff) {
10     this.cnt = (this.cnt + diff);
11   }
12 }
13 ICounter c = new Counter();
14 c.Incr(5);
```

Stack:

PC	...		PC	ret	diff	this
14	...		10	null	5	ref 1

Heap:

1
cnt=0

```
1  interface ICounter {
2    void Incr(int diff);
3  }
4  class Counter : ICounter {
5    private int cnt;
6    public Counter() {
7      this.cnt = 0;
8    }
9    public void Incr(int diff) {
10     this.cnt = (this.cnt + diff);
11   }
12 }
13 ICounter c = new Counter();
14 c.Incr(5);
```

Stack:

PC	...		PC	ret
14	...		10	null

Heap:

1
cnt=5

```
1  interface ICounter {
2    void Incr(int diff);
3  }
4  class Counter : ICounter {
5    private int cnt;
6    public Counter() {
7      this.cnt = 0;
8    }
9    public void Incr(int diff) {
10     this.cnt = (this.cnt + diff);
11   }
12 }
13 ICounter c = new Counter();
14 c.Incr(5);
```

Stack:

PC	c
15	ref 1

Heap:

1
cnt=5

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void Incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 Counter c = new Counter();
11 c.Incr(5);
```

Stack:

PC
1

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void Incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 Counter c = new Counter();
11 c.Incr(5);
```

Stack:

PC
10

Heap:

1
cnt=

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void Incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 Counter c = new Counter();
11 c.Incr(5);
```

Stack:

PC	...		PC	ret	this
10	...		4	null	ref 1

Heap:

1
cnt=

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void Incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 Counter c = new Counter();
11 c.Incr(5);
```

Stack:

PC	...		PC	ret
10	...		4	null

Heap:

1
cnt=0

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void Incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 Counter c = new Counter();
11 c.Incr(5);
```

Stack:

PC	c
11	ref 1

Heap:

1
cnt=0

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void Incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 Counter c = new Counter();
11 c.Incr(5);
```

Stack:

PC	...		PC	ret	diff	this
11	...		7	null	5	ref 1

Heap:

1
cnt=0

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void Incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 Counter c = new Counter();
11 c.Incr(5);
```

Stack:

PC	...		PC	ret
11	...		7	null

Heap:

1
cnt=5

```
1  class Counter {
2    private int cnt;
3    public Counter() {
4      this.cnt = 0;
5    }
6    public void Incr(int diff) {
7      this.cnt = (this.cnt + diff);
8    }
9  }
10 Counter c = new Counter();
11 c.Incr(5);
```

Stack:

PC	c
12	ref 1

Heap:

1
cnt=5

```
1  class Counter:
2    def __init__(self):
3      self.cnt = 0
4    def incr(self,diff):
5      self.cnt = (self.cnt + diff)
6  c = Counter()
7  c.incr(5)
```

Stack:

PC
1

```
1  class Counter:
2    def __init__(self):
3      self.cnt = 0
4    def incr(self,diff):
5      self.cnt = (self.cnt + diff)
6  c = Counter()
7  c.incr(5)
```

Stack:

PC
6

Heap:

1

```
1  class Counter:
2    def __init__(self):
3      self.cnt = 0
4    def incr(self,diff):
5      self.cnt = (self.cnt + diff)
6  c = Counter()
7  c.incr(5)
```

Stack:

PC	...		PC	ret	self
6	...		3	None	ref 1

Heap:

1

```
1  class Counter:
2    def __init__(self):
3      self.cnt = 0
4    def incr(self,diff):
5      self.cnt = (self.cnt + diff)
6  c = Counter()
7  c.incr(5)
```

Stack:

PC	...		PC	ret
6	...		3	None

Heap:

1
cnt=0

```
1  class Counter:
2    def __init__(self):
3      self.cnt = 0
4    def incr(self,diff):
5      self.cnt = (self.cnt + diff)
6  c = Counter()
7  c.incr(5)
```

Stack:

PC	c
7	ref 1

Heap:

1
cnt=0

```
1  class Counter:
2    def __init__(self):
3      self.cnt = 0
4    def incr(self,diff):
5      self.cnt = (self.cnt + diff)
6  c = Counter()
7  c.incr(5)
```

Stack:

PC	...		PC	ret	diff	self
7	...		6	None	5	ref 1

Heap:

1
cnt=0

```
1  class Counter:
2    def __init__(self):
3      self.cnt = 0
4    def incr(self,diff):
5      self.cnt = (self.cnt + diff)
6  c = Counter()
7  c.incr(5)
```

Stack:

PC	...		PC	ret
7	...		6	None

Heap:

1
cnt=5

```
1  class Counter:
2    def __init__(self):
3      self.cnt = 0
4    def incr(self,diff):
5      self.cnt = (self.cnt + diff)
6  c = Counter()
7  c.incr(5)
```

Stack:

PC	c
8	ref 1

Heap:

1
cnt=5

```
1  def f(x):
2    if (x > 0):
3      return (f(-20) + 1)
4    else:
5      return (x * 2)
6  f(20)
```

Stack:

PC
1

```
1  def f(x):
2    if (x > 0):
3      return (f(-20) + 1)
4    else:
5      return (x * 2)
6  f(20)
```

Stack:

PC	...		PC	ret	x
6	...		2	None	20

```
1  def f(x):
2    if (x > 0):
3      return (f(-20) + 1)
4    else:
5      return (x * 2)
6  f(20)
```

Stack:

PC	...		PC	ret	x
6	...		3	None	20

```
1  def f(x):
2    if (x > 0):
3      return (f(-20) + 1)
4    else:
5      return (x * 2)
6  f(20)
```

Stack:

PC	...		PC	...		PC	ret	x
6	...		3	...		2	None	-20

```
1  def f(x):
2    if (x > 0):
3      return (f(-20) + 1)
4    else:
5      return (x * 2)
6  f(20)
```

Stack:

PC	...		PC	...		PC	ret	x
6	...		3	...		5	None	-20

```
1  def f(x):
2    if (x > 0):
3      return (f(-20) + 1)
4    else:
5      return (x * 2)
6  f(20)
```

Stack:

PC	...		PC	...		PC	ret
6	...		3	...		5	-40

```
1  def f(x):
2    if (x > 0):
3      return (f(-20) + 1)
4    else:
5      return (x * 2)
6  f(20)
```

Stack:

PC	...		PC	ret
6	...		4	-39

```
1  def f(x):
2    if (x > 0):
3      return (f(-20) + 1)
4    else:
5      return (x * 2)
6  f(20)
```

Stack:

PC
10

This is it!

The best of luck, and thanks for the attention!