Stack

Discussion G

Java Packages

- Namespace
  - **`a.b.c.X`**
  - **`a.b.c.d.X`**
- Directory Tree Hierarchy
  - **`.../Import/a/b/c/X`**
  - **`.../Import/a/b/c/d/X`**
- Classpath
  - **`.../Import`**

To Postfix

- Copy operands to output
- Push operators on the Stack
- Push higher precedence operator
- Evaluate the higher precedence operator by popping to output
- (lowest precedence
-) lowest precedence

PostFix Form

```
(1 +  2) * (3  + 4)       //6
```

Input	Stack	Output	
(	(		//1
1		1	//2
+	+		//3
2		2	//4
)	===	+	//5
*	*		//6
(	(		//7
3		3	//8
+	+		//9
4		4	//10
)	===	+	//11
		*	//12

Evaluate Postfix

```
1 2 + 3 4 + *            //6
```

Input	Stack	
		//1
1	1	//2
2	2 1	//3
+	3	//4
3	3 3	//5
4	4 3 3	//6
+	7 3	//7
*	21	//8

```
21                                              //9
```

Execution Stack (1)

```
public class Global {                              //1
  private static Stack<String> callstack;
  static {
   callstack = new Stack<String>();
  }
  static void addCall(String name) {
                             callstack.push(name);}
  static void removeCall () {callstack.pop();}
  static void printCallChain () {
    System.err.println("Currently executing chain:");
    for (String s: callstack) {
      System.err.println(s);
    }
    System.err.println("...");
  }
}
```

Execution Stack (2)

```
public class X {                                     //2
  public void methodOne (Y y) { y.methodTwo(); }
  public static main  (String [] args) {
    X x = new X();
    Y y = new Y();
    x.methodOne(y);
  }
}

public class Y {                                     //3
  public void methodTwo () {}
}
```

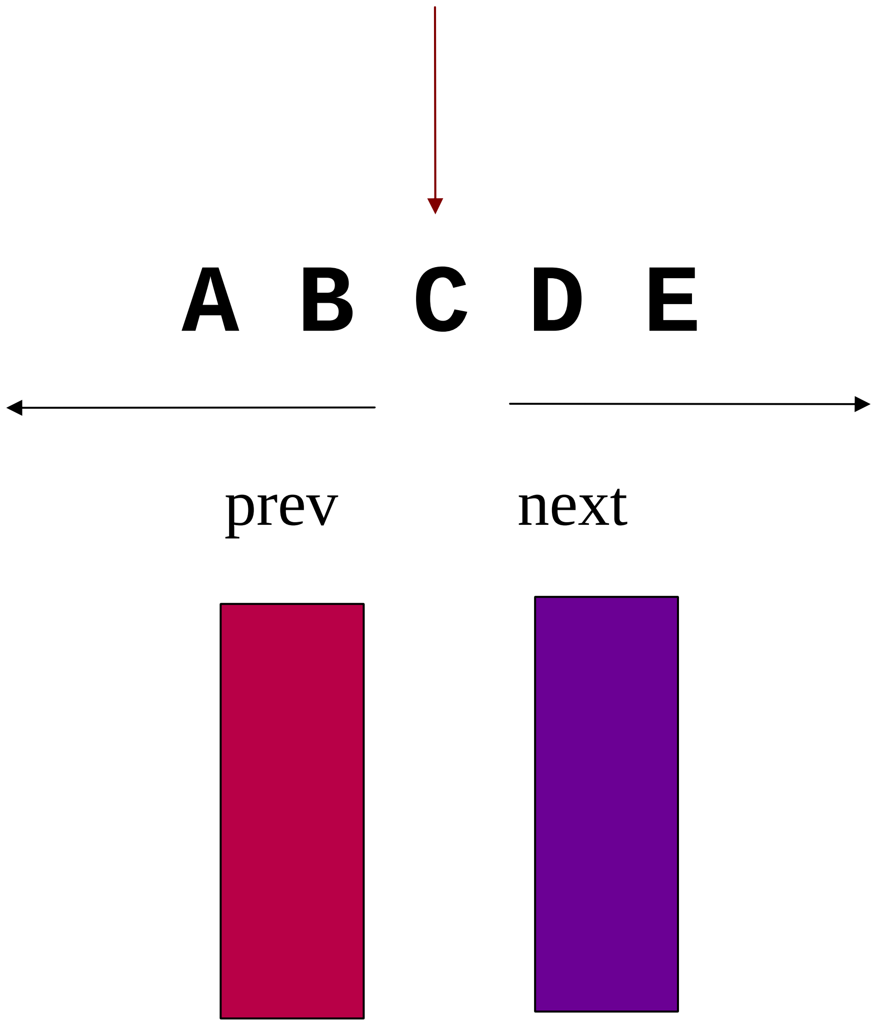
Execution Stack (3)

```
public class X {                                        //2

  public void methodOne (Y y) {

    Global.addCall(“X::methodOne(” + y + ”)”);
    Global.printCallChain();

    y.methodTwo();

    Global.removeCall();
  }

}
```

Execution Stack (4)

```
public class Y {                                        //3

  public void methodTwo () {

    Global.addCall("Y::methodTwo()");
    Global.printCallChain();

    // Method Body


    Global.removeCall();

  }
}
```

History (1)



History (2)

```
public class History {

  private Stack<Page> next;
  private Stack<Page> prev;
  private Page current;

  public History () {
    next = new Stack<Page>();
    prev = new Stack<Page>();
  }

  public Page next () {  }
  public Page previous () {  }
  public void setNext(Page p) {  }
}
```

History (3)

```
public Page next () {
  Page p = next.pop();
  if (current != null)  prev.push(current);
  current = p;
  return p;
}

public Page previous () {
  Page p = prev.pop();
  if (current != null)  next.push(current);
  current = p;
  return p;
}
```

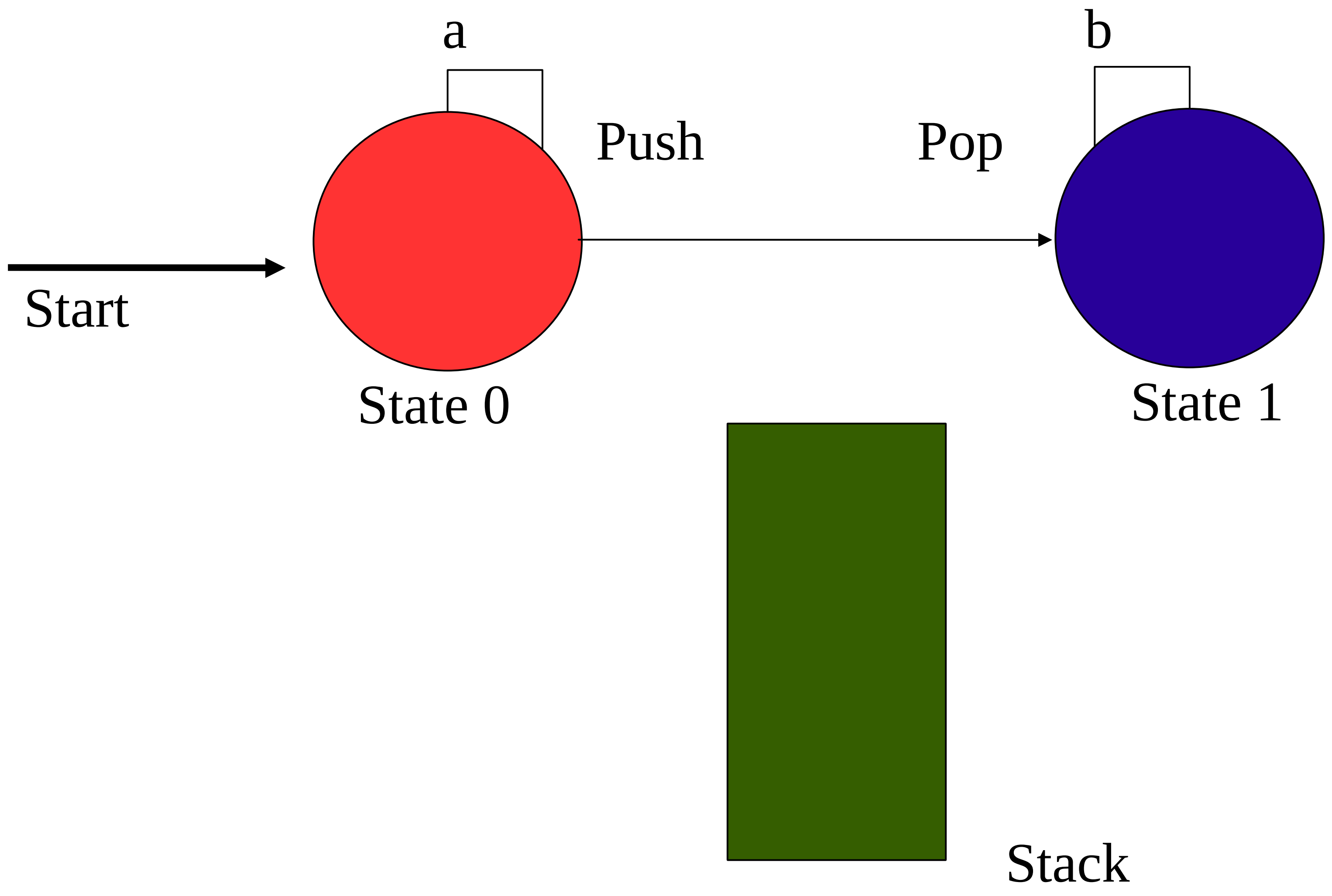
History (4)

```
public void setNext(Page p) {
  if (current != null)  prev.push(current);
  current = p;
}
```

Browser

```
public class Browser {
  private History history;
  public Browser (String homepage) {
    history = new History();
    view(homepage);
  }

  public view (String page) {
    Page p = new Page(page);
    history.setNext(p);
  }
  public void view(Page p) {}
  public void prev(){view(history.previous());}
  public void next(){view(history.next()); }
}
```

Push Down Automaton



Sequence(1)

```
public class SequenceAnBnChecker {

  InputStream is;
  Stack<Character> sc;

  public SequenceAnBnChecker(InputStream is) {
    this.is = is;
    sc = new Stack<Character>();
  }

  public boolean check () {}

    public static void main (String[] args) { }

}
```

Sequence(2)

```
public boolean check () {

  char a, b, r;
  try {

    }
  } catch (IOException ioe) {
    // is okay, end of stream
  } catch (EmptyStackException ese) {
    return false;
  }

  return sc.empty();
}
```

Sequence(3)

```
try {
  a = r = (char) is.read();
  sc.push(a);
  while ( a == (r = (char) is.read()) ) {
    sc.push(a);
  }
  b = r;
  sc.pop();
  while ( b == ( r = (char) is.read()) ) {
    sc.pop();
  }
}
```

Sequence(4)

```
public static void main (String[] args) {

    StringBufferInputStream sb =
               new StringBufferInputStream(args[0]);

    SequenceAnBnChecker anbn =
               new SequenceAnBnChecker(sb);

    boolean pass = anbn.check();
    System.out.println(pass);
  }
```