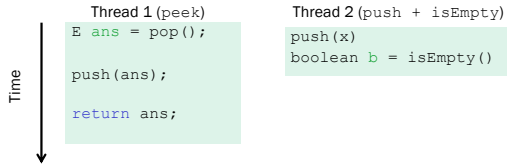
Example of a Race Condition, but not a Data Race

```
class Stack<E> {
  … // state used by isEmpty, push, pop
  synchronized boolean isEmpty() { … }
  synchronized void push(E val) { … }
  synchronized E pop() {
    if(isEmpty())
      throw new StackEmptyException();
    …
  }
  E peek() { // this is wrong
     E ans = pop();
     push(ans);
     return ans;
  }
}
```


8

Example 1: peek and isEmpty

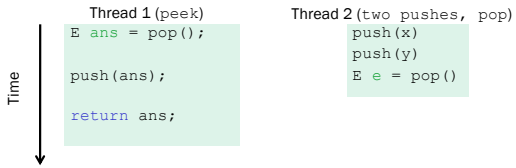
- **Property we want**: If there has been a **`push`** (and no **`pop`**), then **`isEmpty`** should return **`false`**
- With **`peek`** as written, property can be violated – how?



11

Example 2: peek and push

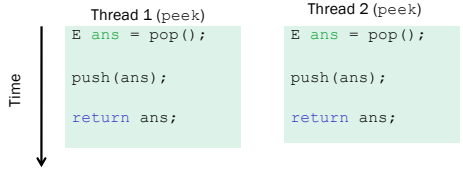
- **Property we want:** Values are returned from **`pop`** in LIFO order
- With **`peek`** as written, property can be violated – how?



13

Example 4: peek and peek

- **Property we want:** **`peek`** doesn't throw an exception unless stack is empty
- With **`peek`** as written, property can be violated – how?



16

Motivating Deadlock Issues

Consider a method to transfer money between bank accounts

```
class BankAccount {
  …
  synchronized void withdraw(int amt) {…}
  synchronized void deposit(int amt) {…}
  synchronized void transferTo(int amt, BankAccount a) {
    this.withdraw(amt);
    a.deposit(amt);
  }
}
```

Potential problems?