Wednesday January 9

Lecture 2

Precondition (service) condition

```
int divide(int x, int y){
    if (y == 0) {
        throw
    }
}
```

throws

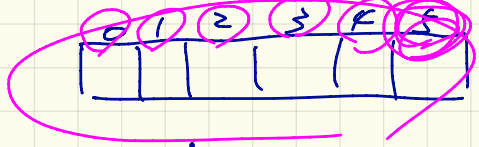
error condition

```
divide(x, y: INTEGER): INT
    require
        y /= 0
```

service condition

binSearch (x , xs)

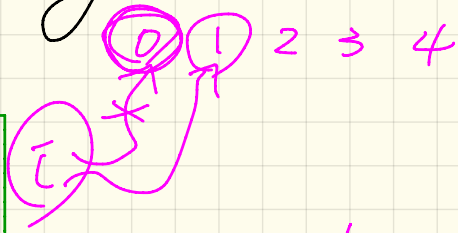
precondition: xs is sorted in non-descending order.



Math

$$\forall i, j \mid 0 \le i, j \le xs.length - 1 \bullet$$
$$i < j \Rightarrow xs[i] \le xs[j]$$

implies



$$\forall i \mid 0 \le i \le xs.length - 2 \bullet$$
$$xs[i] \le xs[i+1]$$

dummy variable

Cursor variable

across 0 |..| xs.count - 2 as i

all

xs[i.item] <= xs[i.item + 1]

end

Bank Accounts in Java: Version 2

```
1  public class AccountV2 {
2    public AccountV2(String owner, int balance) throws
3       BalanceNegativeException
4    {
5      if( balance < 0 ) { /* negated precondition */
6        throw new BalanceNegativeException(); }
7      else { this.owner = owner; this.balance = balance; }
8    }
9    public void withdraw(int amount) throws
10      WithdrawAmountNegativeException, WithdrawAmountTooLargeException {
11     if( amount < 0 ) { /* negated precondition */
12       throw new WithdrawAmountNegativeException(); }
13     else if ( balance < amount ) { /* negated precondition */
14       throw new WithdrawAmountTooLargeException(); }
15     else { this.balance = this.balance - amount; }
16   }
```

Bank Accounts in Java: Version 2 Critique (1) (Compared with Version 1)

```
1  public class BankAppV2 {
2    public static void main(String[] args) {
3      System.out.println("Create an account for Alan with balance -10:");
4      try {
5        AccountV2 alan = new AccountV2("Alan", -10);
6        System.out.println(alan);
7      }
8      catch (BalanceNegativeException bne) {
9        System.out.println("Illegal negative account balance.");
10     }
```

```
Create an account for Alan with balance -10:
Illegal negative account balance.
```

Bank Accounts in Java: Version 2 Critique (2) (Compared with Version 1)

```
1  public class BankAppV2 {
2    public static void main(String[] args) {
3      System.out.println("Create an account for Mark with balance 100:");
4      try {
5        AccountV2 mark = new AccountV2("Mark", 100);
6        System.out.println(mark);
7        System.out.println("Withdraw -1000000 from Mark's account:");
8        mark.withdraw(-1000000);
9        System.out.println(mark);
10     }
11     catch (BalanceNegativeException bne) {
12       System.out.println("Illegal negative account balance.");
13     }
14     catch (WithdrawAmountNegativeException wane) {
15       System.out.println("Illegal negative withdraw amount.");
16     }
17     catch (WithdrawAmountTooLargeException wane) {
18       System.out.println("Illegal too large withdraw amount.");
19     }
```

Console Output:

```
Create an account for Mark with balance 100:
Mark's current balance is: 100
Withdraw -1000000 from Mark's account:
Illegal negative withdraw amount.
```

Bank Accounts in Java : Version 2 Critique (3) (Compared with Version 1)

```
1  public class BankAppV2 {
2    public static void main(String[] args) {
3      System.out.println("Create an account for Tom with balance 100:");
4      try {
5        AccountV2 tom = new AccountV2("Tom", 100);
6        System.out.println(tom);
7        System.out.println("Withdraw 150 from Tom's account:");
8        tom.withdraw(150);
9        System.out.println(tom);
10     }
11     catch (BalanceNegativeException bne) {
12       System.out.println("Illegal negative account balance.");
13     }
14     catch (WithdrawAmountNegativeException wane) {
15       System.out.println("Illegal negative withdraw amount.");
16     }
17     catch (WithdrawAmountTooLargeException wane) {
18       System.out.println("Illegal too large withdraw amount.");
19     }
```

Console Output:

```
Create an account for Tom with balance 100:
Tom's current balance is: 100
Withdraw 150 from Tom's account:
Illegal too large withdraw amount.
```

Bank Accounts in Java: Version 2 Critique (4)

```java
public class AccountV2 {
  public AccountV2(String owner, int balance) throws
      BalanceNegativeException
  {
    if( balance < 0 ) { /* negated precondition */
      throw new BalanceNegativeException(); }
    else { this.owner = owner; this.balance = balance; }
  }
  public void withdraw(int amount) throws
      WithdrawAmountNegativeException, WithdrawAmountTooLargeException {
    if( amount < 0 ) { /* negated precondition */
      throw new WithdrawAmountNegativeException(); }
    else if ( balance < amount ) { /* negated precondition */
      throw new WithdrawAmountTooLargeException(); }
    else { this.balance = this.balance - amount; }
  }
```

Supplier

Fix 1: balance <= amount

```java
public class BankAppV2 {
  public static void main(String[] args) {
    System.out.println("Create an account for Jim with balance 100:");
    try {
      AccountV2 jim = new AccountV2("Jim", 100);
      System.out.println(jim);
      System.out.println("Withdraw 100 from Jim's account:");
      jim.withdraw(100);
      System.out.println(jim);
    }
    catch (BalanceNegativeException bne) {
      System.out.println("Illegal negative account balance.");
    }
    catch (WithdrawAmountNegativeException wane) {
      System.out.println("Illegal negative withdraw amount.");
    }
    catch (WithdrawAmountTooLargeException wane) {
      System.out.println("Illegal too large withdraw amount.");
    }
```

Client

Req:

REQ1: Each account is associated with the *name* of its owner (e.g., "Jim") and an integer *balance* that is always positive.

Console Output:

```
Create an account for Jim with balance 100:
Jim's current balance is: 100
Withdraw 100 from Jim's account:
Jim's current balance is: 0
```

Bank Accounts in Java : Version 3

```
1  public class AccountV3 {
2    public AccountV3(String owner, int balance) throws
3       BalanceNegativeException
4    {
5     if(balance < 0) { /* negated precondition */
6       throw new BalanceNegativeException(); }
7     else { this.owner = owner; this.balance = balance; }
8     assert this.getBalance() > 0 : "Invariant:  positive balance";
9    }
10   public void withdraw(int amount) throws
11      WithdrawAmountNegativeException, WithdrawAmountTooLargeException {
12    if(amount < 0) { /* negated precondition */
13      throw new WithdrawAmountNegativeException(); }
14    else if (balance < amount) { /* negated precondition */
15      throw new WithdrawAmountTooLargeException(); }
16    else { this.balance = this.balance - amount; }
17    assert this.getBalance() > 0 : "Invariant:  positive balance";
18   }
```

0

False

Bank Accounts in Java: Version 3 Critique (1) (Compared with Version 2)

```
1  public class BankAppV3 {
2    public static void main(String[] args) {
3      System.out.println("Create an account for Jim with balance 100:");
4      try { AccountV3 jim = new AccountV3("Jim", 100);
5            System.out.println(jim);
6            System.out.println("Withdraw 100 from Jim's account:");
7            jim.withdraw(100);
8            System.out.println(jim); }
9            /* catch statements same as this previous slide:
10            * Version 2: Why Still Not a Good Design? (2.1) */
```

```
Create an account for Jim with balance 100:
Jim's current balance is: 100
Withdraw 100 from Jim's account:
Exception in thread "main"
    java.lang.AssertionError: Invariant: positive balance
```

Bank Accounts in Java: Version 3 Critique (2)

```
1  public class AccountV3 {
2    public void withdraw(int amount) throws
3        WithdrawAmountNegativeException, WithdrawAmountTooLargeException {
4      if( amount < 0 ) { /* negated precondition */
5        throw new WithdrawAmountNegativeException(); }
6      else if ( balance < amount ) { /* negated precondition */
7        throw new WithdrawAmountTooLargeException(); }
8      else { this.balance = this.balance - amount; }
9      assert this.getBalance() > 0 : "Invariant:  positive balance"; }
```

When amount is neither negative nor too large,

is there any obligation on the supplier of withdraw?

Bank Accounts in Java : Version 4 (with an evil supplier)

```
1  public class AccountV4 {
2    public void withdraw(int amount) throws
3      WithdrawAmountNegativeException, WithdrawAmountTooLargeException
4      if(amount < 0) { /* negated precondition */
5        throw new WithdrawAmountNegativeException(); }
6      else if (balance < amount) { /* negated precondition */
7        throw new WithdrawAmountTooLargeException(); }
8      else { /* WRONT IMPLEMENTATION */
9        this.balance = this.balance + amount; }
10     assert this.getBalance() > 0 :
11       owner + "Invariant: positive balance"; }
```

Bank Accounts in Java: Version 4 Critique

acc.bal precond. 100 balance >150 acc.withdraw (...) acc.bal postcondition

```
1  public class BankAppV4 {
2    public static void main(String[] args) {
3      System.out.println("Create an account for Jeremy with balance 100:")
4      try { AccountV4 jeremy = new AccountV4("Jeremy", 100);
5            System.out.println(jeremy);
6            System.out.println("Withdraw 50 from Jeremy's account:");
7            jeremy.withdraw(50);
8            System.out.println(jeremy); }
9      /* catch statements same as this previous slide:
10      * Version 2: Why Still Not a Good Design? (2.1) */
```

old bal.

new bal.

```
Create an account for Jeremy with balance 100:
Jeremy's current balance is: 100
Withdraw 50 from Jeremy's account:
Jeremy's current balance is: 150
```

balance = old balance − amount