MIDTERM REVIEW

Abstraction Functions

- Internal (like the representation invariant)
- Client doesn't need this!
- Can be used to show code correctness when combined with spec and rep invariant
- Maps concrete representation to abstract value

Abstraction Functions

- AF: *R* ⇒ *A*
- *R:* Set of objects
  - Consists of fields in the class; concrete, code
- A: Set of abstract objects
  - What the object means; abstract, conceptual
- AF:
  - References internal code representation
  - Can contain calculations, etc that the client doesn't care about

Abstraction Functions

```
public class Line {
  private Point start;
  private Point end;
  …}
```

```
// AF(r) = line l such that
// l.start = r.start
// l.end = r.end
```

Abstraction Functions

```
/**
 * Card represents an immutable playing card.
 * @specfield suit: {Clubs,Diamonds,Hearts,Spades}
 * @specfield value: {Ace,2,...,Jack,Queen,King}
 */

public class Card {
    private int index;
    …
}
```

```
// suit = S(index div 13)
// where S(0)=Clubs, S(1)=Diamonds, …
// value = V(index mod 13)
// where V(1)=Ace, V(2)=2, ...,
// V(12)=Queen, V(0)=King
```

Specification strength

- Stronger specification is:
  - Easier or harder for the client to use?
  - Easier or harder for the implementer to specify?
- To weaken a specification, you can:
  - Strengthen or weaken the preconditions?
  - Strengthen or weaken the postconditions?

Documentation

```
class IntegerSet {
      private List<Integer> set = new
LinkedList<Integer>();

      public boolean contains(int x) {
            int index = set.indexOf(x);
            if (index != -1) {
                  set.remove(index);
                  set.add(0, x);
            }
            return index != -1;
      }
}
```

@requires?
@modifies?
@effects?
@return?
@throws?

Backwards Reasoning

$\{ (x * y) * y^{n-1} = b \} \Rightarrow \{ x * y^n = b \}$

`x = x * y;`

$\{ x * y^{n-1} = b \}$

`n = n - 1;`

$\{ x * y^n = b \}$

Forwards Reasoning

$\{ |x| > 2 \}$

`x = x * 2;`

$\{ |x| > 4 \}$

`x = x - 1;`

$\{ x > 3 \mid x < -5 \}$

CoinPile Class

```
class CoinPile {
      private List<Integer> coins;
      public CoinPile() {
            coins = new ArrayList<Integer>();
      }

      ... // many more methods for adding and
      removing coins, computing change, etc.
}
```

CoinPile Class

```
class CoinPile {
      private List<Integer> coins;
      public CoinPile() {
            coins = new ArrayList<Integer>();
      }

      ... // many                    r adding and
      removing co                    ange, etc.
}
```

@specfield pennies: int
@specfield nickels: int
@specfield dimes: int
@specfield quarters: int

- Representation invariant?
- Abstraction function?

CoinPile Class, cont'd

```
@returns a list of coins with one coin of
value n for each coin in this with value n
(i.e., the list of coins in this)

public List<Integer> getCoins() {
      return new ArrayList<Integer>(coins);
}
```

Representation exposure?