Practice with Contracts

Yu-Shan Sun[1] / Kevin Plis[2]

[1]School of Computing
Clemson University
Clemson, SC 29634 USA

[2]Department of Computer Science
University of Vermont
Burlington, VT 05405 USA

CPSC 2150: Wednesday, Sept. 2, 2020

Slide Sources:

1. Previous/Current CPSC 2150 Instructors

Last Class

1. Classes and Objects
2. Invariants
3. Good Practices
4. Types of Objects

Homework Assignment

1. Watch the various videos posted on your lab Canvas page.
2. Programming assignment 1 is posted on your lab Canvas page.

Consider a Speedster Class

- Keeps track of a time and distance, and calculates the velocity
  - As the object keeps moving, we add to the total distance and time
  - Does not track directionality
- Constructor/Methods:
  1. **Constructor:** `Speedster(double d, double t)`
  2. `void addTravel(double d, double t)`
  3. `double getVelocity()`
  4. `double getTotalDistace()`
  5. `double getTotalTime()`
- What are our contracts?

`Speedster` Contract Notes

- Invariants are not repeated in our contracts
  - No postcondition of `Velocity = Distance / Time`
  - We will need to enforce in code
- Some seem obvious
  - Get functions
  - `Distance = d` on constructor
    - Still need to specify
  - Use `@pre` to help enforce invariants
    - Limit the data that can come in

Consider a Mortgage Class

- Decides whether or not to approve a mortgage, etc
- **Needs:**
  - Credit score (0 - 850)
  - Interest rate (APR)
  - Monthly debt payments for customer
  - Annual income
  - Years for the mortgage (5 - 30)
  - Debt to income ratio:
    $((monthlydebt + monthlypayment)/monthlyincome)$
  - Monthly payment
  - House cost
  - Down Payment (must be 3.5% of the cost)
- Constructor/Methods
  1. **Constructor:**
     ```
     Mortgage(int cs, double apr, double debt,
       double income, double cost, double dp)
     ```
  2. `boolean isApproved()`
  3. `double getPayment()`

Consider a `Mortgage` Class

Note:

$$monthly\ payment = \frac{monthly\ interest\ rate * principal}{1 - (1 + monthly\ interest\ rate)^{total\ num\ payments}}$$

- Invariants?

Consider a Mortgage Class

Constructor

```
Mortgage(int cs, double apr, double debt,
  double income, double cost, double dp)
```

- @pre?
- @post?

Consider a `Mortgage` Class

`isApproved`

- Rejects a loan if (credit score < 600 and debt to income ratio > 25%) or if (dtoi > 40%) or if down payment < 3.5%
- `@pre`?
- `@post`?

Consider a Mortgage Class

getPayment

- @pre?
- @post?

Consider Two Methods

```
1 int daysInAMonth(int m, int y)
2 boolean isLeapYear(int y)
```

- **Note:** April, June, September and November have 30 days
- **Leap Year Rules:** If year is divisible by 4, but not divisible by 100 unless also divisible by 400, then it is a leap year

Conclusion

- Contracts can get complicated very quickly
- Difficult to specify formally
  - But we can mix in some information to be helpful
- Contracts depend on our design
  - Is the percent down > 3.5% enforced as a precondition to the constructor, or in determining if the loan is approved?
  - Writing contracts is part of our design process
    - We define what each method does
    - Also defines how they interact and affect the state of the object

Homework Assignment

1. Watch the various videos posted on your lab Canvas page.
2. Programming assignment 1 is posted on your lab Canvas page.

Summary

1. Practice with Contracts