Top-Down Design, Bottom-Up Implementation

Rose-Hulman Institute of Technology
Computer Science and Software Engineering

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY

Plan for Today

- Design, implement, and test Blackjack together
  - Lots of whiteboard work
  - I'll share code after class

+2 extra-credit in-class quiz points if you sign up and attend

-5 in-class quiz points if you sign up and don't attend without informing me before noon on Sunday

Sunday, 3:30-5:30, Olin 257

Software Sunday Sign-up

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

Designing/Implementing a Larger Program

- Most of our programs have been small
- For larger programs, we need a strategy
- One common strategy: *top-down design*
  - Break the problem into a few big pieces
    - One function for each piece
  - Break each piece into smaller pieces
  - Continue until the pieces are "bite size"

Recall: Top-level Algorithm

- Create initial card deck
- Deal initial cards
- Player plays until busted or chooses to stop
- Dealer plays until required to stop
- Report who wins

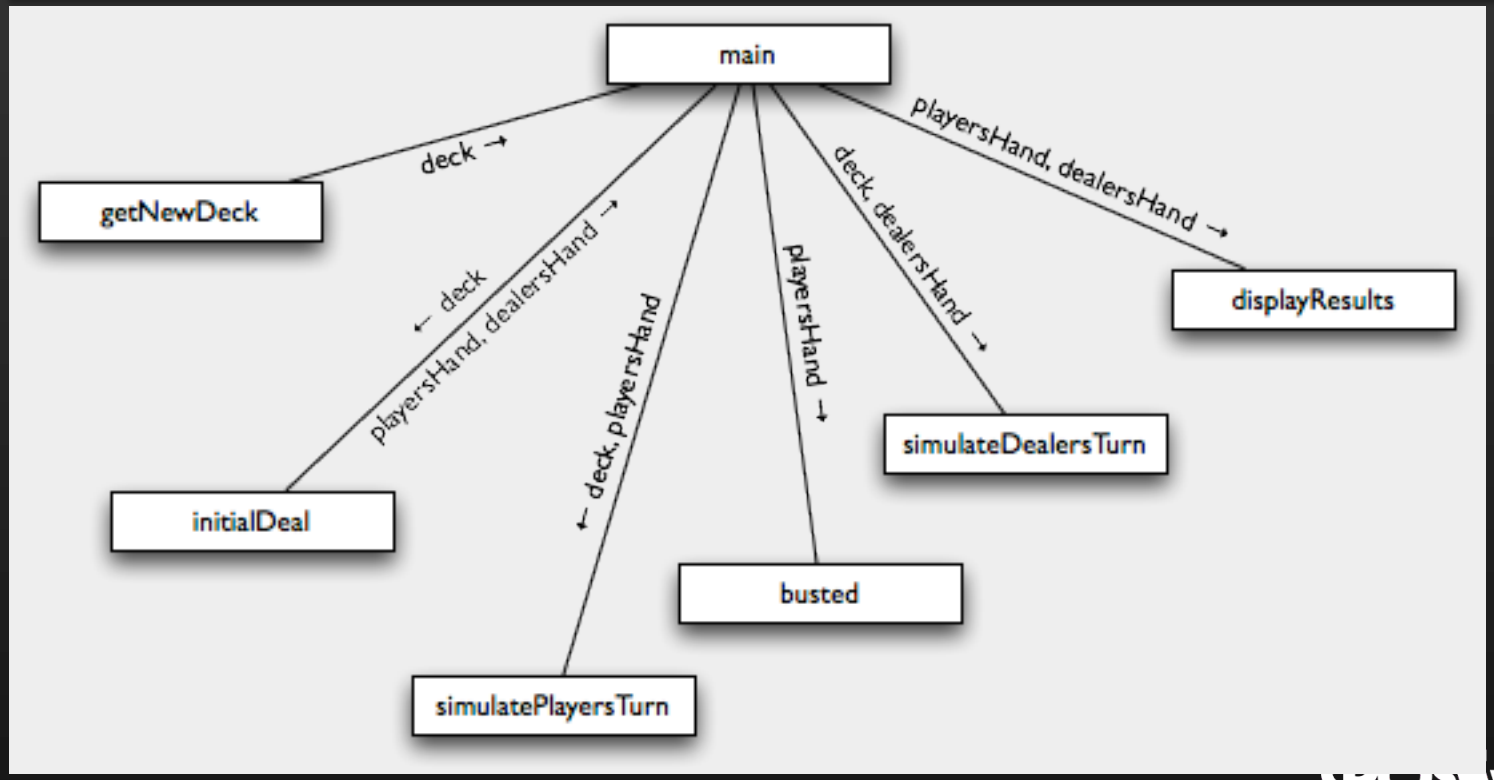
Top-level Functions Called by main

- **getNewDeck()**—Creates and returns a complete deck of cards
- **initialDeal(deck)**—Deals cards from the deck to each player, returns the hands
- **simulatePlayersTurn(deck, playersHand)**—Allows player to choose hit or stay
- **busted(playersHand)**—Checks whether the given hand is over 21
- **simulatePlayersTurn(deck, dealersHand)**—Dealer hits or stays, based on the rules
- **displayResults(playersHand, dealersHand)**—Determines and displays who wins.

Implementation of main

```
def main():
    deck = getNewDeck()
    playersHand, dealersHand = initialDeal(deck)
    simulatePlayersTurn(deck, playersHand)
    if(not busted(playersHand)):
        simulateDealersTurn(deck, dealersHand)
    else:
        print("Busted!!!")
    displayResults(playersHand, dealersHand)
```

Initial Blackjack Structure Diagram



Some Preliminary Data Values

```
# Define some constants used by many functions
suits = ['Clubs', 'Diamonds', 'Hearts', 'Spades']

cardNames = ['Ace', 'Deuce', '3', '4', '5',
             '6', '7', '8', '9', '10',
             'Jack', 'Queen', 'King']

winningScore = 21

dealerMustHoldScore = 16


# Card is represented by a list: [cardName, suit]
# Examples: ['Ace','Clubs'] or ['7','Diamonds']
# A hand or a deck is a list of cards.
```

Bottom-up Testing

- Implement and test as we go
- Small changes, well tested make debugging easy

Class Exercise

Design, Implement, and Test

Q3-4

ROSE-HULMAN
INSTITUTE OF TECHNOLOGY