### **Domain Model for Blackjack**

Based on your description, the domain model for Blackjack should include the following concepts:

* **Game**: Manages the flow of the Blackjack game, including starting new games and rounds, dealing with bets, and determining winners.
* **Round**: Represents a single cycle of play in the game.
* **Player**: Participates in the game, places bets, makes decisions to hit or stand, and holds a hand of cards.
* **Dealer**: A special kind of Player that follows house rules for hitting and stands and deals cards to players.
* **Deck**: A set of playing cards used in the game.
* **Card**: An individual playing card with a rank and may have a suit in a standard deck.
* **Hand**: Represents the set of cards a Player or Dealer has at any point.
* **Bet**: The amount of money wagered by the Player for a round.

Attributes and associations:

* **Game**:
  + Has a Deck
  + Has multiple Players (1 to 9)
  + Has one Dealer
  + Conducts multiple Rounds
* **Round**:
  + Belongs to a Game
  + Involves Players' Bets
* **Player**:
  + Has a Hand
  + Places a Bet
  + Makes decisions to hit or stand
* **Dealer** (extends Player):
  + Deals Cards
  + Plays following house rules
* **Deck**:
  + Consists of Cards
  + Is used to draw Cards for Players and Dealer
* **Card**:
  + Belongs to a Deck
  + Has a rank and suit
* **Hand**:
  + Contains Cards
  + Belongs to a Player or Dealer
* **Bet**:
  + Is placed by a Player
  + Is associated with a Round

### **System Sequence Diagram for a Single Player Starting a New Game**

I will now sketch a system sequence diagram that outlines the interactions between the player and the system when starting a new game and playing one round of Blackjack.

The system sequence diagram will include the following steps:

1. The player indicates the start of a new game.
2. The player places a bet.
3. The system deals two cards to the player and two to the dealer, with one dealer card face down.
4. The player chooses to hit or stand.
5. If the player hits, the system deals another card.
6. This repeats until the player stands or busts.
7. Once the player stands or busts, the dealer reveals the face-down card.
8. The dealer hits until the hand value is greater than 16.
9. The system evaluates the hands and determines the outcome (player wins, loses, or ties).
10. The system settles bets accordingly.

### **Identifying and Fixing Coupling/Cohesion Issues**

Based on the object model you've provided, here are two potential problems and their solutions:

**Problem 1: High Coupling between Game and Deck**

* The Game class directly interacts with the Deck to draw cards.
* **Solution**: Introduce a method in the Dealer class that encapsulates the card drawing logic, which can be called by the Game.

**Problem 2: Low Cohesion in Participant Class**

* The Participant class is responsible for holding cards, scoring, and iterating to the next participant.
* **Solution**: Split the Participant class into two: Player (for holding cards and scoring) and PlayerManager (to iterate over players and manage turns).

### **Object-Level Interaction Diagram for a Player Choosing to Hit**

Finally, for the interaction diagram when a player chooses to hit, the following sequence of events will be depicted:

1. The player sends a "hit" request to the Game.
2. The Game forwards the request to the Dealer.
3. The Dealer draws a Card from the Deck.
4. The Dealer gives the Card to the Player, adding it to the Player's Hand.
5. The Game checks for the Player's score.

### **System Sequence Diagram - Textual Description**

1. **Actor: Player**
   * Initiates the game by selecting "New Game."
2. **System: Game**
   * Responds to the initiation by creating a new game session.
   * Calls the shuffleDeck() method on the Deck object.
3. **Actor: Player**
   * Places a bet by calling the placeBet(betAmount) method on the Game object.
4. **System: Game**
   * Validates the bet amount.
   * Calls the dealInitialCards() method on the Dealer object to start the round.
5. **System: Dealer**
   * Deals two cards to the Player by calling the drawCard() method on the Deck for each card and adds them to the Player's Hand.
   * Deals two cards to themselves, with one card face down.
6. **Actor: Player**
   * Decides to hit or stand. If the player decides to hit, they call the hit() method on the Game object.
7. **System: Game** (Loop for each hit)
   * Calls the dealCardToPlayer() method on the Dealer object to give the Player a new card.
   * Checks if the Player has busted after each hit.
8. **System: Game**
   * Once the Player stands or busts, it calls the playDealerHand() method on the Dealer object.
9. **System: Dealer**
   * Reveals the face-down card.
   * Continues to draw cards until the hand value is greater than 16 by calling drawCard() on the Deck and adding to its own Hand.
10. **System: Game**
    * Evaluates both the Player's and Dealer's hands by calling the determineOutcome() method.
    * Settles the bets based on the outcome.

### **Visual Representation**

Now, to visualize the system sequence diagram, imagine a vertical timeline where the Player's actions are on the left, and the system's responses are on the right. Each step described above corresponds to a horizontal line between the Player and the System, representing the messages exchanged. Return arrows are drawn for responses from the system to the player, where appropriate.

Here's a rough sketch of how these interactions could be visually represented:

