Folder: Game:

Class: IBettingRound

Boolean placeBet(Bet bet) -
Expected: *Check if the gambler doesn't have any other active bets (There can be only one)*

Tests:
- *No_other_active_bets_place_bet_should_return_true*
- *There_are_active_bets_place_bet_should_return_false*

Checked somewhere else
The card placing the bet has enough credit to place the bet amount (this should be checked when the bet is created and not here.)-
Enough_Credit_On_Card_To_Place_Bet_PASS()
Enough_Credit_On_Card_To_Place_Bet_FAIL()

Set<Bet> getAllBetsMade() -
Expected: Gets a Set of all the Bet type objects.
Tests:
- *should_return_0_bet_objects_PASS*
- *should_return_10_bet_objects_PASS*

int numberOFBetsMade() -
Expected: *Returns the number of bets made.*
Tests:
- *Returned_number_should_always_be_equal_or_over_0 **
- *should_return_10_PASS*

Class: IGameRule

BetResult determineWinner(Integer randomWinValue, Set<Bet> bets)-
Expected:*Determine the winner from a set of Bets, using the random win value.*
Tests:
- *should_return_second_cardID_in_list_PASS*
- *should_return_5th_cardID_in_list_PASS*

Int getMaxBetsPerRound() -
Expected: *Can't be less than 1, can't be more than maxInt*
Tests:
- *returned_number_should_always_be_over_0 **
- *should_return_10_PASS*

Class: IGame

Void startBettingRound() -
Expected: *Create and start a new betting round, when called then a new bettinground is active the current*

tops the current active round and creates a new betting round.
Tests:

- *Current_Betting_round_is_removed*
- *New_Started_Betting_round_has_default_values*

boolean isBettingRoundFinished() -
Expected: Several tests

1) *If there is another betting round- stop it.*

boolean acceptBet(Bet bet, IGamingMachine gamingMachine) -

- ***throws NoCurrentRoundException-***

Expected:

1) *If not a current round throws a* ***NoCurrentRoundException***

Folder BBBB:

Class: Bet

Constructor:

Public Bet(BetID betID, MoneyAmount moneyAmount)

- **Expected**: Bet with Id and moneyAmount

Methods:

Folder Gamining_Machine:

Class: IGamingMachine

Boolean **placeBet**(long amountInCents) throws NoPlayerCardException
Expected: ***Accepts param of amount in cents return true if the bet is placed false if this does not happen***

Tests:

- ***amount_on_card_is_0_cents_PASS***
- ***amount_is_negative_should_return_false_PASS***
- ***amount_is_positive_should_return_true_PASS***
- ***amount_on_card_is_64_cents_PASS***

Exception NoPlayerCardException
Expected: ***If the players card is not placed in the machine and there is no ID? Throw exception***

Tests:

- ***Non ?***

Void **acceptWinner**(betResult winResult)
Expected: ***Accepts the BetResult from the winner. Clear all open bets on this machine. When the winner has made his bet in this machine: let the cashier update the amount***

Tests:

- ***bet_placed_by_winner_should_be_accepted_PASS***

- ***bet_placed_by_nonWinner_should_not_be_accepted_PASS***
- ***amount_off_bets_on_machine_should_be_0_PASS***

Void **connectedCard**(IPlayerCard Card);
Expected: ***Connect card to this gaming machine***

Tests:
- ***should_return_card_id_PASS***

Folder Cashier:

Class: ICashier

IPlayerCard **distributeGamblerCard**()
Expected: ***New card should be distributed bank teller keeps track of the distributed cards.***
Tests:
- ***Non***

Void **returnGambleCard**(IPlayercard card)
Expected: ***When handing in the card at a Bank teller, all betID's on it are logged. The total amount of money credit is physically handed to the gambler, the amount stored on the card is changed to zero. The stored betID's on the card are also removed.***

Tests:
- ***total_betIds_should_be_10_PASS***
- ***expected_amount_of_money_on_card_should_be_zero_PASS***
- ***betids_should_be_empty_PASS***

Boolean **checkIfBetIdIsValid**(IPlayerCard card, Bet betToCheck) **throws** BetNotExceptedException

Expected: ***check if Bet made with the playercard is possible. this is based on the amount related to the card, and the amount made in the bet.If the bet is valid, the amount of the bet is subtracted from the amount belonging to the card.***

Tests:

- ***insufficient_funds_should_return_false_PASS***
- ***sufficient_funds_should_return_true_PASS***
- ***bet_is_under_zero_return_false_PASS***

Void **addAmount**(IPlayerCard card, MoneyAmount amount)

Expected: ***should add an amount to the players card no negative amounts are allowed***

Tests:

- ***add_negative_amount_CardAmount_should_be_zero_PASS***
- ***cardAmount_should_be_10_PASS***

Class : IPlayerCard

Set<BetID> returnBetIDs();

Expected: ***returns all generated betID's by this card return a copied set of betID's generated by this card.***

Tests:

- ***should_return_10_betIDs_PASS***
- ***should_return_0_betIDs_PASS***

Set<BetID> returnBetIDsAndClearCard();

Expected: ***returns all generated betID's by this card, and clears all betID's from the card. return a copied set of betID's generated by this card.***

Tests:

- ***should_return_10_betIDs_PASS***
- ***should_return_0_betIDs_PASS***
- ***remaining_betIds_after_methodcall_should_be_0_PASS***

BetID generateNewBetID();
Expected: ***The card generates a unique betID for every bet made by the gambler on the machine. A list of all generated betID's is also stored on the card. BetID's also contain a timestamp.***

Tests:

- ***timestamp_added_PASS***
- ***should_return_10_betIDs_PASS***
- ***no_duplicate_bets_on_card_PASS***

int getNumberOfBetIDs();
Expected: ***return number of betID's generated on this card***

Tests:

- ***should_return_0_betIDs_PASS***
- ***should_return_10_betIDs_PASS***

CardID getCardID();

<u>Class</u> : ICasino (and bet)

boolean checkIfBetIsValid(IPlayerCard card, Bet betToCheck) - card exists, betToCheck exists, betToCheck is from this card
Expected: true if valid/ false all other cases