**Folder: Game:**

**Class: IBettingRound- if we can not mock it (doesn’t have access to id)**

***Boolean* *placeBet(Bet bet)* -**

**Expected: *Check if the gambler doesn’t have any other active bets (There can be only one) place the current bet to ‘bet’***

***Requirements:***

*Player bets.count are 0 placing a bet is possible.*

*Player bets.count are more than 0 and placing a bet is NOT possible.*

*Player ‘bets’ object is null placing a bet is possible.*

( Player money is enough bet is possible

Player doesn’t have enough money placing a bet is not possible.) only checked in ICashier as he is doing that check.

**Tests:**

* ***No\_other\_active\_bets\_place\_bet\_should\_return\_true***
* ***There\_are\_active\_bets\_place\_bet\_should\_return\_false***

**Class: IGameRule**

***BetResult determineWinner(Integer randomWinValue, Set<Bet> bets)-***

***Expected: Determine the winner from a set of Bets, using the random win value.***

***Requirements:***

*Random win value is between 0 and 5 inclusive*

*Must select one random winner from the finished betting round*

**Tests:**

* ***should\_return\_second\_cardID\_in\_list\_PASS***
* ***should\_return\_5th\_cardID\_in\_list\_PASS***

**Class: IGame**

***Void startBettingRound() -***

***Expected: Create and start a new betting round, when called then a new bettingRound is active the current stops the current active round and creates a new betting round.***

***Requirements:***

The current betting round is removed.

A new betting round with default values has been started.

Players > 0

**Tests:**

* ***Current\_Betting\_round\_is\_removed***
* ***New\_Started\_Betting\_round\_has\_default\_values***

***boolean isBettingRoundFinished() -***

***Expected: Determine if the right number of bets are done (determined by gamerules) to be able to calculate a winner.***

***Requirements:***

*Number of bets (players) is more than 5 then finish*

*The betting round has been active for 20 seconds- then finish*

***boolean acceptBet(Bet bet, IGamingMachine gamingMachine) -***

***Expected: If not a current round throws a NoCurrentRoundException***

***Requirements:***

*gamingMachine.Round != null*

*gamingMachine.Round is null throw NoCurrentRoundException*

***Tests:***

* ***Throw\_Exception\_If\_There\_Is\_No\_Current\_Betting\_Round***

**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***

***Requirements:*** *Amount in cents is less than or equal to 0 placing is not possible*

*Amount in cents is positive number placing a bet is possible*

*If card is valid placing a bet is possible.*

*If card is not valid throw* NoPlayerCardException

1. *if\_ID\_is\_null\_*

**Tests:**

* ***Passed\_Amount\_Is\_More\_Than\_Zero***
* ***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 \****

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***

***Requirements:***

Clear all bets

**Tests:**

* ***bet\_placed\_by\_winner\_should\_be\_accepted\_PASS***
* ***bet\_placed\_by\_nonWinner\_should\_not\_be\_accepted\_PASS***
* ***amount\_of\_bets\_on\_machine\_should\_be\_0\_PASS***
* ***winner\_has\_made\_bet\_amountOfBets\_should\_be\_1\_PASS***

Void **connectedCard**(IPlayerCard Card);

**Expected: *Connect card to this gaming machine***

***Requirements:***

Card is valid

Card is added to the list of active cards.

**Tests:**

* ***should\_return\_card\_id\_PASS***

**Folder Cashier:**

**Class:** ICashier

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.***

***Requirements:***

*Money in card are set to 0 after execution.*

*Stored Ids on card are set to 0 after execution.*

**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)

**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.***

***Requirements:***

*Should check the funds found on the card are higher than the placed bet, if so the amount from the bet is subtracted from the total amount from the players 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***

***Requirements:***

*The totalAmount on the playersCard is increased by amount*

**Tests:**

* ***add\_negative\_amount\_CardAmount\_should\_be\_zero\_PASS***
* ***cardAmount\_should\_be\_10\_PASS***

**Class : IPlayerCard**

**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.***

***Requirements:***

*Returns all betId from Set<BetID>*

*Set<BetID> is cleared*

**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.***

**Requirements:**

Uses IDFactory to generate a unique ID

Stores the id in a list of generated ID’s

**Tests:**

* ***timestamp\_added\_PASS***
* ***should\_return\_10\_betIDs\_PASS***
* ***no\_duplicate\_bets\_on\_card\_PASS***

**CardID getCardID();**

**Class : ICasino (and bet)**

**boolean checkIfBetIsValid(IPlayerCard card, Bet betToCheck)**

**Requirements:**

*Card != null*

*betToCheck != null*

**Expected: true if valid/ false all other cases**

**Won’t be tested:**

1. ***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***

1. **int getNumberOfBetIDs();**

**Expected: *return number of betID's generated on this card***

**Requirements:**

**Tests:**

* ***should\_return\_0\_betIDs\_PASS***
* ***should\_return\_10\_betIDs\_PASS***

1. **Set<BetID> returnBetIDs();**

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

***Requirements:***

**Tests:**

* ***should\_return\_10\_betIDs\_PASS***
* ***should\_return\_0\_betIDs\_PASS***

1. ***int numberOFBetsMade() -***

***Expected: Returns the number of bets made.***

**Tests:**

* ***Returned\_number\_should\_always\_be\_equal\_or\_over\_0 \****
* ***should\_return\_10\_PASS***

1. IPlayerCard **distributeGamblerCard**()

**Expected: *New card should be distributed bank teller keeps track of the distributed cards.***

***Requirements:***

New card is created.

1. ***Int getMaxBetsPerRound()* -**

**Expected: *returns 5***

**Tests:**

* ***returned\_number\_should\_always\_be\_over\_0 \****
* ***should\_return\_10\_PASS***