

Roulette Presentation

MC Student
January 12, 2007

This is essentially a text version of the javadoc I created within my program files. But it serves as a nice explanation of what the project does.

Player.java =====

Player.java creates player objects with the variable object components:
name, cash, stillPlaying, betType, betAmount, betNumber, everWantInsurance,
and hasInsurance.

This class sets and retrieves these values, and disables play when a player runs out of cash or opts out of further play.

After a bet is made and won or lost, this class resets the player object's values to default values, except for name, cash, and everWantInsurance, which perpetuate throughout play. After losing a bet, Player.java evaluates whether the player is eligible to continue play based on the amount of cash the player has remaining, and disables further play as needed by setting stillPlaying to false.

There is only one constructor method, and it is called during game setup by Roulette.java.

DEFAULT VALUES:

cash	(int)	0
stillPlaying	(boolean)	true
betType	(int)	1
betAmount	(int)	0
betNumber	(int)	0
everWantInsurance	(boolean)	false
hasInsurance	(boolean)	false

BET TYPES:

betType(1):	betting a specific number
betType(2):	betting on high numbers
betType(3):	betting on low numbers
betType(4):	betting on even numbers
betType(5):	betting on odd numbers

INSURANCE:

The "additional feature" I've added into my project is the option of purchasing insurance. If a player has chosen to be prompted for insurance, they may purchase it before any wheel spin for a set price, \$INSURANCE. If the wheel lands on zero, the player does not automatically lose their money. Rather, they only lose the cost of insurance; their original bet amount is returned to their playing cash balance. This is described further in the following section, Roulette.java.

Roulette.java =====

Roulette.java simulates a game of roulette at a casino. Each player is an object as defined in Player.java. Player.java is called upon to create new player objects, to set object variables, and to affect implications of winning/losing bets and resetting bet default parameters after a bet is completed.

Players are permitted to choose the amount of money they bring to the table, but cannot add to that amount during the game except by winning bets. Players are permitted to continue playing as long as they still have money to gamble. Once out of money, the player can no longer play until a new game is started. Similarly, once a player voluntarily opts out of playing, the player cannot resume betting until a new game is initiated.

BETTING:

The roulette wheel has values of and including 0 through 36. If the player wants to place a bet, they can do so by choosing the bet type from these options: high (19 - 36), low (1 - 18), even (2, 4, 6, ...), odd (1, 3, 5, ...), and number (guess the actual number that will come up in a wheel spin, 1 - 36). Each turn the player can bet any sum of money greater than zero and less than or equal to their total present cash balance. A player cannot bet on 0.

WINNING:

A player wins the bet if they correctly predict the type or value of the number coming up on the next spin of the wheel. If a player bets on even/odd or high/low and wins, the player is paid double the amount of their bet. If the player bets on a number and the number comes up, the player is paid ten times the amount of their initial bet.

ZERO:

A player cannot bet on the number zero coming up on the wheel. If the number zero does come up, the player automatically loses the bet and is not paid. Insurance is offered at \$INSURANCE cost per spin, and in the event a zero comes up, if a player has purchased insurance on the spin at hand, the player is refunded the bet amount, but forfeits the cost of the insurance.

Game Flow through the Project =====

A welcome message first appears and lets the user know that they will first be prompted for player names, then additional information will be requested of them. Once finished entering names, the user can type 'done' to signal that there will be no more names.

The user is then prompted for each of the names of players who will be playing. The default MAX_PLAYERS is 10, but this can be increased or decreased as desired by editing the Roulette.java code.

After entering the desired names, the user enters 'done' or reaches the MAX_PLAYERS limit and is no longer signaled to enter names.

The Player objects are then created using the constructor and setter methods contained in Player.java. An array of Players holds references to each of these objects.

Next, the user is prompted to enter data for each player name: starting cash and whether or not the player should be prompted to buy insurance before each bet.

"The rules" are then displayed, describing to the players how bets may be placed and what the payout is for each type. It warns about the zero and sells the players on insurance again.

A quick check from a boolean routine `anyPlayersStillPlaying()` will start cycling through the `playGame()` routine, which calls on helper methods `determineBetAmount()`, `determineBetType()`, and `spinTheWheel()`.

The betting begins. Each player in turn is asked how much they'd like to bet, and if they've opted to be prompted to buy insurance, that is offered to them. Then the player chooses the type of bet, and the wheel spins. A random number is generated in `spinTheWheel()`, from 0 to `MAX_WHEEL_VAL` (set to 36). `spinTheWheel()` goes through most of the actual calculation work that takes place in this game, analyzing whether or not the player has lost. It checks for insurance in the event that a zero was spun, and once determining the outcome win, lose, or had insurance, calls on the appropriate method in `Player` to redistribute money, refund the bet, reset player variables back to default, and set whether or not the player wishes to continue.

A check is in place so that if a player runs out of money after a losing bet, they are automatically unable to play another round and are so warned.

Interesting Clips =====

Here is an example of both insurance at work, as well as setting the `MAX_WHEEL_VAL` to 2 so we can expect a lot of 0's to come up...

```

                *   *   *   *   *   *   *   *   *   *   *   *   *   *
Melissa, you have $10995. How much would you like to bet? $990
Would you like to buy insurance?
y
Melissa, what type of bet do you want to place? Please choose from this list:

Number, Even, Odd, High, or Low
=   =   =   =   =
Your choice: lo

Spinning the wheel.   .   .   .   .   .   0

    *** Aren't you glad you got insurance!!***

Do you want to continue playing?
```

And here is an example of running out of cash. I like that I warn players who might otherwise be interested in purchasing insurance that they have insufficient funds, so I don't ask them if they want it...

```

                *   *   *   *   *   *   *   *   *   *   *   *   *   *
Melissa, you have $1970. How much would you like to bet? $1970
You don't have enough left to buy insurance.
Melissa, what type of bet do you want to place? Please choose from this list:

Number, Even, Odd, High, or Low
=   =   =   =   =
Your choice: low

Spinning the wheel.   .   .   .   .   .   0

LOSER! Sorry, you lose.
What's more, Melissa, you're broke now. No more spins for you.
```

Thanks for playing!