# DoubleDice ver.3

|  |  |
| --- | --- |
| ver0 | Original description |
| ver1 | Finding class, attribute and method candidates |
| ver2 | Description refining the use of DoubleDice |
| ver3 | Refining the logic, attributes and methods. |
| ver4 | Tkinter implementation, class inheritance and composition |

# Original Description

# Double dice, a game of betting the result of throwing two dice. First, a user places his or her bet, 1...100 points and makes the guess: the sum is less, equal or bigger than 7 e.g. 1+6, 2+5, 3+4. The program throws the dice. And, if the sum is less or bigger and the player guessed it correctly he wins the bet \* 100 points. If the sum is 7 and the user had guessed that he wins the bet \* 1000 points. But if the sum was bigger when the guess was smaller or wise versa, the user loses bet \* 10 points. And if the guess was equal and the sum is not 7 he loses bet \* 100 points. The points are saved from each game. There are no negative points, but the points are set to zero.

# 1st iteration, check nouns

Double dice, a game of betting the result of throwing two dice. First, a user places his or her bet, 1...100 points and makes the guess: the sum is less, equal or bigger than 7 e.g. 1+6, 2+5, 3+4. The program throws the dice. And, if the sum is less or bigger and the player guessed it correctly he wins the bet \* 100 points. If the sum is 7 and the user had guessed that he wins the bet \* 1000 points. But if the sum was bigger when the guess was smaller or wise versa, the user loses bet \* 10 points. And if the guess was equal and the sum is not 7 he loses bet \* 100 points. The points are saved from each game. There are no negative points, but the points are set to zero.

|  |  |  |  |
| --- | --- | --- | --- |
| noun | basic form, synonym, duplicate value, notes | value or collections | possible class, object, attribute |
| Double dice |  | one value, name of the game, title | game’s title |
| game |  | object, on run of the game inside the program | game, possible class, knows the rules of the game and keeps track of the game’s data |
| result | sum | one value, sum of two dice numbers. It is a sum of collection though. | attribute to dice |
| dice |  | object, gets thrown. | dice, possible class, makes the result to be guessed. |
| user, player |  | object, has points | possible class, can own points |
| points |  | one value, the amount of “score” you can get or lose by betting | attribute to user |
| program |  | program to drive the game object |  |
| bet |  | value of points that is put into the game | possible attribute to the game class |
| guess |  | value if the result is going to be </>/= of 7 | possible attribute to the game |
| zero |  | Minimum amount of score user can have | possible attribute to user as a minimum score threshold |

# 2nd iteration, check verbs.

Double dice, a game of betting the result of throwing two dice. First, a user places his or her bet, 1...100 points and makes the guess: the sum is less, equal or bigger than 7 e.g. 1+6, 2+5, 3+4. The program throws the dice. And, if the sum is less or bigger and the player guessed it correctly he wins the bet \* 100 points. If the sum is 7 and the user had guessed that he wins the bet \* 1000 points. But if the sum was bigger when the guess was smaller or wise versa, the user loses bet \* 10 points. And if the guess was equal and the sum is not 7 he loses bet \* 100 points. The points are saved from each game. There are no negative points, but the points are set to zero.

|  |  |  |
| --- | --- | --- |
| verb | subject-object(who does, who is the target) | possible action, function, method |
| is | Double dice is a game |  |
| betting | User bets game takes the bet | method for user input getting the bet |
| throw | game throws the dice | method for initializing the result |
| makes the guess | user makes guess if result is </>/= of 7 | method for initializing the guess |
| wins | user wins points, if the guess is correct | possible method for the game |
| loses | user loses points, if the guess is wrong | possible method for the game |
| saved | points are saved from each game | possible method for the game |
| are set to zero | points are set to zero if all points are lost, no negative values | possible method for the game |

# 3rd iteration, specification ver1,ver2 and ver3

### @ver3 description

Double dice, is a game that is played with two dice. At the beginning of the game, two dice are thrown but are not shown to the player. Player starts the game with 100 points. After the dice are thrown, the player needs to place their bet which is always between 1 and 100 points. Then the player is asked if the sum of the dice is less, equal, or bigger than 7. After being asked, the player makes their guess by selecting one of the options listed, "less", "equal" or "greater". If the player's guess and the sum are in the same range, the player gets its bet \* 100 points, e.g., player guesses "less" and the sum of the dice comes under 7. If the player guesses "equal", and the sum of the dice is 7 then the player gets its bet \* 1000 points but if the sum of the dice is not 7, then the player loses its bet \* 100 points. If the sum of the dice and the player's guess are in opposite ranges, then the player loses its bet \* 10. Overall points for the game cannot be negative and in case this happens the points are set to zero. The points are saved for each game.

## Description

ver3 description added with changes in ver2

### Attributes that are removed with @ver3:

Roll, value, points

### Attributes that are added with @ver3:

title, rules, dice\_sum, both\_dice, player\_guess, player\_points, initial\_points

name, id, points 🡪 Player class

### Methods that are added with @ver3

place\_bet, make\_guess à Player class

# Description

1. Double dice is a class that has:
   1. Attributes
      1. title
         1. @ver3 Type: String, Title of the game.
      2. dice1
         1. Type: Integer, values between 1 and 6
      3. dice2
         1. Type: Integer, values between 1 and 6
      4. both\_dice
         1. Type: Tuple, stores the values of dice1 and dice2
      5. dice\_sum
         1. Type: Integer, values between 2 and 12
      6. bet\_value
         1. Type: Integer, values from 0 to 100
         2. @ver3 attribute name changed from betvalue to bet\_value
      7. player\_guess
         1. @ver3 Type: String, options: less, equal or greater
      8. win
         1. Type: Boolean, true if player won the bet
      9. Initial\_points
         1. Type: Integer, starting points for the player which is 100.
      10. player\_points
          1. Type: Integer, the amount of points player has
          2. @ver3 attribute name changed from points to player\_points
      11. rules
          1. @ver3 Type: string, shows the rules for the game.
   2. Methods
      1. throw
         1. Assigns random numbers from 1 to 6 for each dice
      2. bet
         1. Inputs players bet value [1 - 100]
      3. guess
         1. Inputs players guess value [2 - 12]
      4. sum
         1. Adds values of dice1 and dice2 and returns the sum.
      5. result
         1. Compares the sum and guess of the player and calculates the new points.
      6. save\_points
         1. Saves the result to points
      7. restart
         1. Restarts the game
      8. run
         1. Runs the game
2. Player is class that has:
   1. Attributes
      1. name
         1. Stores the name/nickname of the player
      2. id
         1. Stores an integer value for the player id
      3. Points
         1. Stores the points the player receives during a game
   2. Methods
      1. place\_bet
         1. Inserts the desired bet points between range 1 - 100
      2. make\_guess
      3. Reserved methods
      4. Reserved methods

Graphical user interface, application

Description automatically generated

# Generated UML class diagram (<https://app.genmymodel.com/>) @ver3

# Code generated from the UML diagram @ver3

class DoubleDice(object):

def \_\_init\_\_(self):

self.title = ""

self.dice1 = 0

self.dice2 = 0

self.both\_dice = None

self.dice\_sum = 0

self.bet\_value = 0

self.player\_guess = ""

self.win = False

self.initial\_points = 0

self.player\_points = 0

self.rules = ""

# Start of user code -> properties/constructors for DoubleDice class

# End of user code

def throw(self):

# Start of user code protected zone for throw function body

return 0

# End of user code

def bet(self):

# Start of user code protected zone for bet function body

return 0

# End of user code

def guess(self):

# Start of user code protected zone for guess function body

return ""

# End of user code

def sum(self):

# Start of user code protected zone for sum function body

return 0

# End of user code

def result(self):

# Start of user code protected zone for result function body

return False

# End of user code

def save\_points(self):

# Start of user code protected zone for save\_points function body

raise NotImplementedError

# End of user code

def restart(self):

# Start of user code protected zone for restart function body

raise NotImplementedError

# End of user code

# Start of user code -> methods for DoubleDice class

# End of user code

class Player(object):

def \_\_init\_\_(self):

self.name = ""

self.id = 0

self.points = 0

# Start of user code -> properties/constructors for Player class

# End of user code

def place\_bet(self):

# Start of user code protected zone for place\_bet function body

return 0

# End of user code

def make\_guess(self):

# Start of user code protected zone for make\_guess function body

return ""

# End of user code

# Start of user code -> methods for Player class

# End of user code

# Start of user code -> functions/methods for DoubleDice package

# End of user code

# Completed code @ver3

# File: doubledice.py

# Author(s): XXXXXX XXXXXX, XXXXXX XXXXXX, XXXXXX XXXXXX, XXXXXX XXXXXX

# CO-Author(s): Pavel Kaljunen, Sebastian Sopola, Uras Ayanoglu, Jerry Karkainen

# Description: This is a dice game called that checks player guess if it is less, equal or greater than 7 against outcomes of the dice.

# --------------------------------------------------------------------------------------------------------------------------------------

# Import necessary libaries

import random

# --------------------------------------------------------------------------------------------------------------------------------------

class DoubleDice:

def \_\_init\_\_(self):

self.title = "Double Dice"

self.dice1 = 0

self.dice2 = 0

self.both\_dice = tuple()

self.dice\_sum = 0

self.bet\_value = 0

self.player\_guess = None

self.win = False

self.initial\_points = 100

self.player\_points = 100

self.rules = """

Double Dice Game Rules:

- Double dice, is a game that is played with two dice. Player starts the game with 100 points.

At the beginning of the game, two dice are thrown but are not shown to the player.

After the dice are thrown, the player needs to place their bet which is always between 1 and 100 points.

- Then the player is asked if the sum of the dice is less, equal, or bigger than 7.

After being asked, the player makes their guess by selecting one of the options listed, "less", "equal" or "greater".

1. If the player's guess and the sum are in the same range, the player gets its bet \* 100 points.

(e.g., player guesses "less" and the sum of the dice comes under 7.)

2. If the player guesses "equal", and the sum of the dice is 7 then the player gets its bet \* 1000 points

but if the sum of the dice is not 7, then the player loses its bet \* 100 points.

3. If the sum of the dice and the player's guess are in opposite ranges, then the player loses its bet \* 10.

(e.g., player guesses "less" and the sum of the dice come over 7.)

- Overall points for the game cannot be negative and in case this happens the points are set to zero.

The points are saved for each game.

"""

@property

def player\_points(self):

return self.\_\_player\_points

@player\_points.setter

def player\_points(self,new\_value):

self.\_\_player\_points = new\_value

@property

def bet\_value(self):

return self.\_\_bet\_value

@bet\_value.setter

def bet\_value(self, new\_value):

self.\_\_bet\_value = new\_value

@property

def dice\_sum(self):

return self.\_\_dice\_sum

@dice\_sum.setter

def dice\_sum(self, new\_value):

self.\_\_dice\_sum = new\_value

@property

def win(self):

return self.\_\_win

@win.setter

def win(self, new\_value):

self.\_\_win = new\_value

@property

def both\_dice(self):

return self.\_\_both\_dice

@both\_dice.setter

def both\_dice(self, new\_value):

self.\_\_both\_dice = new\_value

@property

def dice1(self):

return self.\_\_dice1

@dice1.setter

def dice1(self, new\_value):

self.\_\_dice1 = new\_value

@property

def dice2(self):

return self.\_\_dice2

@dice2.setter

def dice2(self, new\_value):

self.\_\_dice2 = new\_value

# Start of user code -> properties/constructors for DoubleDice class

# End of user code

def throw(self):

# Start of user code protected zone for throw function body

self.dice1 = random.randint(1, 6)

self.dice2 = random.randint(1, 6)

self.both\_dice = (self.dice1, self.dice2)

return self.both\_dice

# End of user code

def bet(self):

# Start of user code protected zone for bet function body

bet\_value = input(f"\nPlace a bet between 1 and 100 points: ")

while not (bet\_value.strip().isnumeric() and 1 <= int(bet\_value.strip()) <= 100):

print("Wrong type or Wrong interval!")

bet\_value = input(f"\nPlace a bet between 1 and 100 points: ")

self.bet\_value = int(bet\_value)

return self.bet\_value

# End of user code

def guess(self):

# Start of user code protected zone for guess function body

guess\_options = ["less","equal","greater"]

self.player\_guess = input("Please type in your guess among options 'less, greater or equal': ").strip().lower()

if self.player\_guess not in guess\_options:

print("Please enter a valid guess!")

return self.guess()

else:

return self.player\_guess

# End of user code

def sum(self):

# Start of user code protected zone for sum function body

self.dice\_sum = self.both\_dice[0] + self.both\_dice[1]

return self.dice\_sum

# End of user code

def result(self):

# Start of user code protected zone for result function body

if self.dice\_sum == 7 and self.player\_guess == "equal":

print("You win!")

self.win = True

self.player\_points += (self.bet\_value \* 1000 - self.bet\_value)

elif self.dice\_sum != 7 and self.player\_guess == "equal":

print("You lose!")

self.win = False

self.player\_points -= (self.bet\_value \* 100 + self.bet\_value)

elif self.dice\_sum < 7 and self.player\_guess == "less":

print("You win!")

self.win = True

self.player\_points += (self.bet\_value \* 100 - self.bet\_value)

elif self.dice\_sum > 7 and self.player\_guess == "greater":

print("You win!")

self.win = True

self.player\_points += (self.bet\_value \* 100 - self.bet\_value)

elif self.player\_points < 0:

print("Your overall points are negative now. We're setting it back to 0.")

self.player\_points = 0

else:

print("You lose!")

self.win = False

self.player\_points -= (self.bet\_value \* 10 + self.bet\_value)

print(f"You now have {self.player\_points} points.")

return self.win

# End of user code

def save\_points(self):

# Start of user code protected zone for save\_points function body

pass

# End of user code

def restart(self):

# Start of user code protected zone for restart function body

self.dice1 = 0

self.dice2 = 0

self.both\_dice = tuple()

self.dice\_sum = 0

self.bet\_value = 0

self.player\_guess = ""

self.player\_points = self.initial\_points

# End of user code

def run(self):

self.throw() # returns game.both\_dice --> tuple (dice1, dice2)

self.bet() # returns game.bet\_value --> int

self.sum() # returns the sum of the dices --> int value

self.guess() # returns game.player\_guess --> string, less, equal, greater

self.result()

print(f"Thrown dice are {self.both\_dice[0]}, {self.both\_dice[1]}. Sum of dice is {self.both\_dice[0] + self.both\_dice[1]}")

# Start of user code -> methods for DoubleDice class

# End of user code

# Start of user code -> functions/methods for minigames lab1 Double dice package

# End of user code