### Example 2. Refining Guess-the-number ver2

|  |  |
| --- | --- |
| ver0 | original description |
| ver1 | finding class, attribute and method candidates |
| ver2 | description refining the use of Guess-the-number |
| ver3 | extending the use and refining the public interface |

## Original description

Guess-the-number is a game where the machine generates one integer between the minimum and the maximum e.g. 1..9. The player inputs a number which is checked if it is the same as the magic number. If it is, the user wins. Or if the guess was too low or high the program shows the information and asks to guess again. When the user inputs the correct answer, the script shows the correct number and how many times the player guessed before hitting the correct value. When the game is over, the user can select to play again.

## 1st iteration, check nouns

**Guess-the-number** is a **game** where the **machine** generates one **integer** between the **minimum** and the **maximum** e.g. 1..9. The **player** inputs a **number** which is checked if it is the same as the **magic** **number**. If it is, the **user** wins. Or if the **guess** was too low or high the **program** shows the **information** and asks to guess again. When the **user** inputs the correct **answer** the **script** shows the correct **number** and how many times the **player** guessed before hitting the correct **value**. When the **game** is over, the **user** can select to play again.

|  |  |  |  |
| --- | --- | --- | --- |
| noun | basic form, synonym, duplicate value, notes | value or collections | possible class, object, attribute |
| **Guess-the-number** |  | one value, name of the game, title | game’s title |
| **game** |  | object, one run of the game inside the program | game, possible class, knows the rules of the game and keeps track of the game’s data |
| **machine, program, script** | program, script | program to drive the game object |  |
| **integer** | number, magic number |  | game knows the magic number, type integer |
| **minimum** | range(min, max) |  | game knows the range from where it generates the magic number |
| **maximum** |  |  |  |
| **player, user** |  |  |  |
| **number, answer, guess** | user’s input, type integer |  | game compares the number to magic number, possibly knows the guess or all the guesses or the number of guesses |
| **information** | result of checking | text in user interface |  |

## 2nd iteration, check verbs

**Guess-the-number** *is* a **game**. **Guess-the-number** *knows* a **magic number** which *is* an **integer** *generated from* a **range** from **minimum** to **maximum** e.g. 1..9. The **program** *asks* **player** *to input* an **integer number** as his **guess**. The **program** *passes* the **guess** to the **Guess-the-number** *to check* if **it** and the **magic number** *are equal*. If they *are equal*, the **player** *wins* and the **program** *shows* the **magic number** and how many guesses the **player** *had*. If the **guess** and **magic number** *are not equal*, the **program** *shows* if the guess *is* too high or too low and *asks* *to guess* again. When the **Guess-the-number** **game** *is over*, the **player** can select to *play again*.

|  |  |  |
| --- | --- | --- |
| verb | subject – object (who does, who is the target) | possible action, function, method |
| is | Guess-the-number is an attribute |  |
| knows | Game knows its attributes |  |
| generated from | Game’s action, generate a random number | generate method, or initialize magic number with a randomly generated number |
| ask, to input | ask, prompt, for input, in the driving program |  |
| pass | pass as an argument to a function (game’s method) |  |
| to check…are … are not equal | game’s operation | check magic number and parameter guess value for equality, returns True or False |
| shows | how many guesses the player had | check adds each guess to attributes (number of guesses or guessed values) |
| asks to guess again | loop, ask until the guess is correct |  |
| play again | loop to play the game (generate magic number, asks for guesses, check and show the result) | driving program  game needs to be re-initialized, restarted (new magic number, reset guesses) |

## 3rd iteration, specification ver1 and ver2

### @ver2 description

Guess-the-number is a game that knows a range of numbers where one or more numbers are randomly picked as magic numbers. The range and number of randomly generated numbers as well as a title, which indicates which game’s rules to follow, are given when creating an object. When a player inputs a guess, it is checked if it is equal to the magic number or numbers, whether the guess is correct or not. All guesses are recorded. In addition to checking if an individual guess is correct Guess-the-number has a method to test if the game is over i.e. all the magic numbers are guessed.

## 4th iteration, specification ver3

Guess-the-number can be used to generate unique numbers or one number from a range e.g. guess a number between 1..9 or to generate several numbers from a range e.g. select three numbers from a range of 1..52. Numbers to generate must be at least one and less than the length of the range. If generated numbers are not unique e.g. generate 2 numbers from 1..6, then the numbers to generate have to be one or bigger. Generated magic numbers can be read but not set from outside the class. For the purposes of different games checking the guesses is done either as if one guess is in the generated numbers, if all the generated numbers have been guessed or if a given number differs from the sum of generated numbers.

### Description

ver1 description added with changes in ver2

1. Guess-the-number is a class that has
   1. attributes (it knows):
      1. title, tells which rules a game object follows
      2. magic number
         1. type: integer @ver2. list
         2. value: generated from a range from minimum to maximum
         3. @ver2. length of the list: numofnums
         4. @ver3 property returns \_\_magic
         5. @ver3 magic.setter raises ValueError
      3. numrange: minimum and maximum, values depend on the game rules @ver2 range -> numrange
         1. @ver2. type: range
         2. @ver2. initial value given as parameter
      4. guesses, how many guesses do player use or actual guesses too?
         1. type: integer @ver2. dictionary
         2. value: initial 0, then only positive numbers @ver2. initial value {'correct':set(), 'wrong':set()}
      5. @ver2. numofnums
         1. type: integer
         2. value: 1 .. number of items in range
         3. initial value given as parameter
         4. @ver3 property returns \_\_numofnums
         5. @ver3 numofnums.setter checks value >= 1 and if unique < len(numrange)
   2. methods
      1. \_init\_\_, initialize all attributes
      2. restart, that initializes magic number and guesses
      3. check, that
         1. has one integer parameter
         2. updates the guesses @ver2. adds the guess to either correct or wrong set
         3. returns the result of the comparison between the parameter and the magic number @ver2. returns True if guess is in magic and False if not
      4. @ver2. isover, that returns True if all magic numbers have been guessed correctly
      5. @ver3 checksum, that returns the difference between sum of magic numbers and given num

### Generated UML class diagram (app.mygenmodel.com) @ver1

Kuva, joka sisältää kohteen teksti

Kuvaus luotu automaattisesti

### Code generated from the UML diagram @ver1

class GuessTheNumber(object):

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

self.title = ""

self.magic = 0

self.guesses = 0

self.min = 0

self.max = 0

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

# End of user code

def check(self):

# Start of user code protected zone for check function body

return False

# 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 GuessTheNumber class

# End of user code

### Completed code ver0

import random

class GuessTheNumber:

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

self.title = "Guess-the-number"

self.min = 1

self.max = 9

self.magic = random.randint(self.min, self.max)

self.guesses = 0

def check(self, guess):

return guess == self.magic

def restart(self):

self.magic = random.randint(self.min, self.max)

self.guesses = 0

### Generated UML class diagram (app.mygenmodel.com) @ver2

Kuva, joka sisältää kohteen pöytä

Kuvaus luotu automaattisesti

### Generated UML class diagram (app.mygenmodel.com) @ver3

### Kuva, joka sisältää kohteen pöytä Kuvaus luotu automaattisesti