An automated system for conducting exams has a class Question which is the parent class for every question type (MCQ/Short/Long). Then each type of question has its own class. For instance, vocabulary questions are represented by the Vocabulary class and code questions are represented by the Script class. In this question, you are required to implement a simplified version of two classes from the this system. You will implement the base class Question and the subclass MCQ, which is used for multiple choice questions. The attributes for these classes are as follows:

**Question**

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Invariant** | **Category** |
| USED\_INDICES | list of all active question indices (ints) | Class attribute |
| \_index | an int > 0 | Immutable instance attribute |
| \_text | a nonempty string | Mutable instance attribute |

**MCQ** (in addition to those inherited)

|  |  |  |
| --- | --- | --- |
| **Attribute** | **Invariant** | **Category** |
| \_choices | nonempty tuple of strings | Mutable instance attribute |
| \_correct | int and a valid index of \_choices | Mutable instance attribute |

Note that a *valid index* can be negative, as \_choices[-1] is the last element of \_choices.

You are required to do the following:

1. Fill in the missing information in each class header.

2. Add any necessary class attributes

3. Add getters and setters as appropriate for the instance attributes

4. Fill in the parameters of each method (beyond the getters and setters)

5. Implement each method according to the specification

6. Enforce any preconditions in these methods using asserts

For methods other then getters and setters, pay attention to the provided specifications. The only parameters are those in the preconditions. The class MCQ may not use any attribute or getter/setter inherited from Question. It may only use super() to access overridden methods.

(a) **The class Question**

class Question # Fill in missing part

"""A class representing a question in the lab system

Attribute USED\_INDICES: A CLASS ATTRIBUTE list of all question indices. This list starts off empty, as there are no questions to start with."""

# ATTRIBUTE \_index: The question index. An int > 0 (IMMUTABLE)

# ATTRIBUTE \_text: The question wording. A nonempty string (MUTABLE)

# CLASS ATTRIBUTE. NO GETTERS OR SETTERS.

# DEFINE GETTERS/SETTERS/HELPERS AS APPROPRIATE. SPECIFICATIONS NOT NEEDED.

def \_\_init\_\_ # Fill in missing part

"""Initializes a new question for the given index and text.

No question can share the same index as another. On creation, the question index is added to the class attribute USED\_INDICES.

Precondition: index is an int > 0, and not already in use.

That is, index cannot be an element of USED\_INDICES.

Precondition: text is a non-empty string"""

def \_\_str\_\_ # Fill in missing part

"""Returns a string representation of this question.

The format is '<index>. <text>'.

Example: If index is 2 and the text is 'What is your quest?',

this method will return '2. What is your quest?' """

def \_\_eq\_\_ # Fill in missing part

"""Returns True if self and other are equal. False otherwise.

An object is equal to this one (self) if it has the same type and the same index. You do not need to compare text, since indices are unique.

Precondition: NONE. other can be ANYTHING """

**(b) The class MCQ**

class MCQ # Fill in missing part

"""A class representing a multiple choice question."""

# ATTRIBUTE \_choices: The options. A nonempty tuple of strings. (MUTABLE)

# ATTRIBUTE \_correct: The index of the correct answer. An int. (MUTABLE)

# ADDITIONAL INVARIANT: \_correct is a valid index of \_choices at all times

# HINT: This allows \_correct to be negative as long as it is in bounds

# DEFINE GETTERS/SETTERS/HELPERS AS APPROPRIATE. SPECIFICATIONS NOT NEEDED.

def \_\_init\_\_ # Fill in missing part

"""Initializes a new multiple choice question with given choices.

Precondition: index is an int > 0, and not already in use.

That is, index cannot be an element of USED\_INDICES in Question.

Precondition: text is a non-empty string

Precondition: choices is a nonempty tuple of strings

Precondition: correct is an int and a valid index of choices

(OPTIONAL ATTRIBUTE; correct is -1, the last choice, by default)"""

def \_\_str\_\_ # Fill in missing part

"""Returns a string representation of this multiple choice question.

The format is '<index>. <text> <answer>'.

For example, suppose the question with index 2 and text 'What is your quest?' has choices ('To pass this exam.', 'To seek the Holy Grail.'). If correct is 1, then the string is '2. What is your quest? To seek the Holy Grail.' """