**Assignment 3 Assessment sheet—**

You are to submit 3 files:

1. The NetBeans class library project that contains your question class library (All the Question classes and QuestionList).
2. A JAR file with your updated Question library.
3. The NetBeans project that contains the main class QuestionsTester which tests your library.

**Evaluation**

Your submission will be evaluated based on the following criteria:

|  |  |  |
| --- | --- | --- |
| **Criteria** | **Mark** | **Comments** |
| **QuestionType Enumeration** | | |
| Constants defined correctly with String value for type description. | /1 |  |
| Constructor - correct name, private modifier, parameter for type field, sets field member to param. | /1.5 |  |
| Type field member with accessor method (correct method header, returns type field value). | /1.5 |  |
| **Total** | **/4** |  |

|  |  |  |
| --- | --- | --- |
| **QuestionType Field in Question Classes** | | |
| **Criteria** | **Mark** | **Comments** |
| Defined with proper modifier in Question | /1 |  |
| Set to correct enum value in all TFQuestion & SAQuestion constructors | /1 |  |
| **Total** | **/2** |  |

|  |  |  |
| --- | --- | --- |
| **MCQuestion Class** | | |
| **Criteria** | **Mark** | **Comments** |
| Task 1.1 & 1.2: questionType field assigned an enum constant | /1 |  |
| Task 2: setOptions() method   * Correct method header * Check for no elements in parameter (throws IAE with concise error message) * Check that no list parameter elements are empty (throws IAE with concise error message) * arrayCopy() valid list parameter to options member | /4.5 |  |
| Task 3: getOptions() method   * Correct method header * Returns copy of options array (not reference to options array - uses arraycopy()) | /2 |  |
| Task 4: isCorrect() override   * Correct method header * Checks that param is String or Character class * Compares guess parameter as char to answer member * Efficient and doesn't use unnecessary if's or comparisons * Returns correct boolean values in all branches | /2.5 |  |
| Task 5: toString() override   * Correct method header * Uses parent toString() for re-usability * Checks for null options array and adds required string to return value * Returns formatted multiple-choice question as specified in the comments/requirements | /3 |  |
|  | **/13** |  |

|  |  |  |
| --- | --- | --- |
| **QuestionList Class** | | |
| **Criteria** | **Mark** | **Comments** | |
| Class header: child of ArrayList, uses concrete Question type | /1 |  | |
| Constructors:   * Default constructor * Single-param constructor has var-arg parameter, adds each object in param to array list * Constructor method headers are correct | /2.5 |  | |
| presentQuestion() method:   * Check for valid index, returns null object if invalid * Returns string as specified in comments/requirements * Re-uses toString() | /2.5 |  | |
| **Total** | **/6** |  | |

|  |  |  |
| --- | --- | --- |
| **QuestionTester Program** | | |
| **Criteria** | **Mark** | **Comments** |
| Task 1.4: displayTypes() method:   * Iterates through enum values() * Display menu using enums as specifed in comments/requirements * Re-uses toString() | /2 |  |
| Task 1.5: returns a QuestionType enum for the user's choice | /1 |  |
| Task 2.2: getQuestion() method:   * Switch/selection to examine type; efficient * Gets answer of appropriate type from user; ensures user can't enter a blank answer * For MC: invokes getMcOptions() method to get options array before getting question answer * Constructs appropriate Question object | /4.5 |  |
| Task 2.4: create the array variable to contain the list of options | /1 |  |
| Task 2.5: gets a valid, non-empty string and add as array element | /1 |  |
| Task 2.6: return array of options | /.5 |  |
| Task 3.0: add new question object to QuestionList | /.5 |  |
| Task 4.1: serveQuiz() method:   * Check for empty list, concise error message * Iteration through QuestionList * Invoke presentQuestion() and get user's guess * Determine if guess is correct or incorrect * Keep track of correct answers * Output score in a professional way * Code is efficient | /4.5 |  |
| **Total** | **/15** |  |

|  |  |  |
| --- | --- | --- |
| **General (applies to entire assignment)** | | |
| **Criteria** | **Mark** | **Comments** |
| **Programming Style/Standards:** Code meets all the industry standards laid out in the [Submission Standards for PROG24178](http://www-acad.sheridanc.on.ca/~jollymor/submissions.html), including, but not limited to:   * Code is fully documented, including required programmer ID documentation, javadocs where appropriate (OOP only), and full internal program documentation. * Code uses proper indentation and spacing, line length, tab sizes, consistent brace style, etc. * Package name meets industry standard or the course package name is used. | /5 |  |
| **Misc.** Penalties will be given for various items, including, but not limited to:   * Submission instructions not followed (0 grade), not a valid NetBeans project (-50% of assignment value), missing components, etc. * Techniques discussed in class have been not been used, failure to meet program requirements. * Program crashes while being tested by professor (-50% of assignment value). | -- |  |
| Late Penalty: -10% per day, up to 3 days |  |  |
| **Assignment Total:** | **/45** |  |