BREAKTHROUGH!

**Skeleton Code Breakdown**

## **Class: Breakthrough**

| **Identifier / Data** | | **Description** |
| --- | --- | --- |
| <<constructor>> | | |
| **Parameters** | n/a | Initialises several private attributes including   * Deck to a new CardCollection * Hand to a new CardCollection * Sequence to a new CardCollection * Discard to a new CardCollection * Score to 0 * GameOver to False * Locks to an empty list * CurrentLock to an empty Lock * LockSolved to False   Invokes the LoadLocks() method to load the external locks text file ‘locks.txt’. |
| **Return values** | n/a |
|  |  |
| AddDifficultyCardsToDeck (private) | | |
| **Parameters** | n/a | Adds five DifficultyCards to the Deck. |
| **Return values** | n/a |
|  |  |
| CheckIfLockChallengeMet (private) | | |
| **Parameters** | n/a | Iterates through the Sequence CardCollection concatenating together the string SequenceAsString with a comma and a space as the separator between each card description.  As a new element from Sequence is concatenated onto the end of SequenceAsString, the string is compared with the Challenge conditions using the CheckIfConditionMet() method on the current lock to check whether a challenge has been met. This is tested incrementally because challenges can be different lengths. If a challenge has been met, True is returned, otherwise False is returned. |
| **Return values** | Boolean |
|  |  |
| CheckIfPlayerHasLost (private) | | |
| **Parameters** | n/a | Checks to see if there are any cards left in the Deck. If there are none, an appropriate message is displayed on the screen together with the final score; the game is over and the method returns True.  If there are cards still left in the Deck, the player has not lost yet, and False is returned, allowing the player to continue playing. |
| **Return values** | Boolean |
|  |  |
| CreateStandardDeck (private) | | |
| **Parameters** | n/a | Used by the SetupGame() method to populate an empty Deck with the correct File, Pick and Keys for each toolkit.  5 Picks from toolkits a, b and c are added to the Deck and then 3 Files and 3 Keys from toolkits a, b and c are also added. |
| **Return values** | n/a |
|  |  |

| **Identifier / Data** | | **Description** |
| --- | --- | --- |
| GetCardChoice (private) | | |
| **Parameters** | n/a | Used by the PlayGame() method to ask the player which card in their Hand they would like to use.  Contains error handling to catch non-integer user input but does not catch data out of range. |
| **Return values** | Value : Integer |
|  |  |
| GetCardFromDeck (private) | | |
| **Parameters** | CardChoice : Integer | Used to get the next card from the Deck CardCollection and add it to the Hand.  If the Deck CardCollection has at least one card in it, the system will then check if the card at position zero in the Deck is a DifficultyCard. If a DifficultyCard is found, the user is asked if they would like to lose a ‘Key’ card or discard the next 5 unseen cards from the Deck. The DifficultyCard is then moved to the Discard CardCollection and the Process() method is invoked on the DifficultyCard passing the user’s Choice as one of the parameters.  The system then performs a check which occurs when repopulating the Hand with cards following a card being played. If another Difficulty card is found during this process, the Difficulty card (or cards if there is more than one sequentially in the Deck) is moved automatically into the Discard CardCollection rather than into the player’s Hand.  If the Deck runs out of cards, the game ends. |
| **Return values** | n/a |
|  |  |
| GetChoice (private) | | |
| **Parameters** | n/a | Used by the PlayGame() method to ask the player if they would like to use a card from their Hand or display the current Discard CardCollection on the screen. |
| **Return values** | Choice : String |
| GetDiscardOrPlayChoice (private) | | |
| **Parameters** | n/a | Used by the PlayGame() method to ask the player if they would like to play the selected card from their Hand to the Sequence or Discard the selected card from their Hand to the Discard CardCollection. |
| **Return values** | Choice : String |
|  | |
| GetRandomLock (private) | | |
| **Parameters** | n/a | Returns a randomly selected lock from the private attribute Locks. |
| **Return values** | Lock |
|  |  |
| LoadGame (private) | | |
| **Parameters** | FileName : String | Uses the FileName parameter to load an external Game text file. Imports the current Score, Challenges, and CardCollections for the Hand, Sequence, Discard and Deck.  True is returned if the file is loaded and processed correctly. If an error occurs, an error message is displayed and False is returned. |
| **Return values** | Boolean |
|  |  |

| **Identifier / Data** | | **Description** |
| --- | --- | --- |
| LoadLocks (private) | | |
| **Parameters** | n/a | Uses a hard-coded ‘locks.txt’ file which contains the locks available for the game. Each line in the text file contains the challenges for a single lock. Each line from the file is split into a string list – Challenges, using a semicolon as a delimiter.  Each Challenge is then further split using a comma as a delimiter into the Conditions for that challenge. The Conditions are then added to a temporary Lock variable – LockFromFile, which is then added to the private attribute Locks list for this game.  If an error occurs, an error message is displayed to advise that the locks.txt file has not loaded correctly. |
| **Return values** | n/a |
|  |  |
| MoveCard (private) | | |
| **Parameters** | FromCollection : CardCollection  ToCollection : CardCollection  CardNumber : Integer | Moves a card at the position of CardNumber from the CardCollection FromCollection to the CardCollection ToCollection.  If the FromCollection is the player Hand and the ToCollection is the Sequence and a valid card has been chosen (i.e. not out of range), the player’s score is updated appropriately for the card being played. For all other moves from one collection to another, Score is not updated.  Score is returned. |
| **Return values** | Score : Integer |
|  |  |
| PlayCardToSequence (private) | | |
| **Parameters** | CardChoice : Integer | This method is used to move a card from the Hand to the Sequence to test it against a lock challenge.  The system tests to see if the Sequence has at least one card in the CardCollection. If it does, the system then checks to see if the card being played by the user is a different ToolType as the previously played card. If the ToolTypes do not match, the card can be played and the card is moved from the Hand to the Sequence and the Score is updated appropriately for that card ToolType. The system then gets a new card from the Deck to put into the Hand.  If the Sequence does not currently have any cards in it, the system moves the chosen card to the Sequence and the Score is updated appropriately.  The system then uses the CheckIfLockChallengeMet() method to confirm if the new card added to the Sequence allows a Challenge to be met and if so displays an appropriate message on the screen and increases the player Score by 5 points. |
| **Return values** | n/a |
|  |  |

| **Identifier / Data** | | **Description** |
| --- | --- | --- |
| PlayGame (public) | | |
| **Parameters** | n/a | This contains the main game loop.  Checks to confirm if the private list attribute Locks contains any locks loaded by the LoadLocks() method. If none have been loaded an error is displayed on the screen and the program quits.  If the list does contain locks, it initialises the following private attributes:   * LockSolved to False * Invokes the SetupGame() method to set up the game   The main game loop runs while the private attribute of GameOver is False. There is then an inner loop which runs while GameOver is False and the private attribute LockSolved is also False.  The inner game loop displays the current user score, the conditions of the current lock and the contents of the Hand, and Sequence CardCollections.  Using the GetChoice() method to display a choice menu to the user, the game loop then uses selection to either display the Discard CardCollection or use a card in the game.  If the user selects to use a card, the system uses the GetCardChoice() method to select a card. It then uses the GetDiscardOrPlayChoice() method to confirm if the user wants to play or discard the chosen card. If the user selects discard, the system moves the selected card from the Hand to the Discard CardCollection and gets a new card from the Deck using GetCardFromDeck(). If the user selects play, the system uses the PlayCardToSequence() method to move the chosen card from the Hand to the Sequence CardCollection.  Once a card has been played or discarded, the main game loop uses the GetLockSolved() method on the CurrentLock to test to see if all the lock challenges have been met. If they have, the LockSolved attribute is set to True and a new lock is generated.  If a lock has been solved, the inner loop returns back to the main game loop which checks if the game is over by invoking the CheckIfPlayerHasLost() method. If this returns True the game ends. |
| **Return values** | n/a |
|  | |
| ProcessLockSolved (private) | | |
| **Parameters** | n/a | Increments the Score by 10 and displays the user score on the screen.  Uses an indefinite loop to iterate through the Discard CardCollection returning all of the cards back to the Deck.  Reshuffles the Deck using the Shuffle() method and assigns a new lock using the GetRandomLock() method with the private attribute CurrentLock. |
| **Return values** | n/a |
|  |  |

| **Identifier / Data** | | **Description** |
| --- | --- | --- |
| SetupCardCollectionFromGameFile (private) | | |
| **Parameters** | LineFromFile : String  CardCol : CardCollection | Used for processing lines 4 to 7 of the external save game file which are for processing the contents of CardCollections (namely the Deck, Discard, Hand and Sequence).  Receives a single line of text (using the LineFromFile parameter) from the external game file as it is imported and processes it into a CardCollection. If the received LineFromFile contains text, it is split into a list of strings – SplitLine, using the comma as the delimiter.  The SplitLine list is then processed iteratively to identify the card number and card type in each element and add it to a CardCollection. If a DifficultyCard is found, that is added instead of a normal ToolCard. |
| **Return values** | n/a |
|  |  |
| SetupGame (private) | | |
| **Parameters** | n/a | Called from the PlayGame() method, this displays the first message of the game on the screen, asking if the player would like to load in an external game file or play a new game. If the player chooses to load the external file the system attempts to load the file ‘game1.txt’. If the file cannot be loaded the game quits.  If the player chooses to play a new game, the system generates a new Deck using the CreateStandardDeck() method and then shuffles it by invoking the Shuffle() method. It then moves 5 cards from the Deck to the Hand to start the player off. The system then invokes the AddDifficultyCardsToDeck() method to add 5 DifficultyCards into the Deck and then reshuffles it again to ensure they are in random locations. The system then assigns a new lock at random to the private attribute CurrentLock using GetRandomLock(). |
| **Return values** | n/a |
|  |  |
| SetupLock (private) | | |
| **Parameters** | Line1 : String  Line2 : String | Used for processing lines 2 and 3 of the external save game file which contain the challenges for the lock.  The parameter Line1 contains line 2 from the external file and the parameter Line2 contains line 3 of the external file. Each line is split into a string list using a semicolon as the delimiter.  The Line1 parameter is then further split using a comma as the delimiter to add a new challenge to the CurrentLock. A single line may contain multiple challenges. The Line2 parameter is split using a semicolon as the delimiter to populate the Met status for each challenge using the SetChallengesMet() method. |
| **Return values** | n/a |
|  |  |

## **Class: Challenge**

| **Identifier / Data** | | **Description** |
| --- | --- | --- |
| <<constructor>> | | |
| **Parameters** | n/a | Initialises the following protected attributes:   * Met to False * Conditions to an empty list |
| **Return values** | n/a |
| GetCondition (public) | | |
| **Parameters** | n/a | Returns a list of strings of the Conditions for this challenge in the lock. |
| **Return values** | Condition : List (String) |
| GetMet (public) | | |
| **Parameters** | n/a | Returns the value of the protected attribute: Met. |
| **Return values** | Met : Boolean |
| SetCondition (public) | | |
| **Parameters** | NewCondition : List (String) | Sets the value of the protected string list attribute: Condition from the parameter NewCondition. |
| **Return values** | n/a |
| SetMet (public) | | |
| **Parameters** | NewValue : Boolean | Sets the value of the protected attribute: Met from the parameter NewValue. |
| **Return values** | n/a |

## **Class: Lock**

*This class does not have a specific constructor and therefore uses the default constructor*

| **Identifier / Data** | | **Description** |
| --- | --- | --- |
| <<constructor>> | | |
| **Parameters** | n/a | Initialises the Challenges protected attribute to an empty list. |
| **Return values** | n/a |
| AddChallenge (public) | | |
| **Parameters** | Condition : List (String) | Initialises a new challenge and sets the value of its condition from the parameter Condition.  Appends the new challenge to the Challenges protected attribute. |
| **Return values** | n/a |
|  |  |
| CheckIfConditionMet (public) | | |
| **Parameters** | Sequence : String | Returns True and sets the challenge to Met by calling SetMet() if the Sequence matches any unsolved challenge, otherwise it returns False. |
| **Return values** | Boolean |
| ConvertConditionToString (private) | | |
| **Parameters** | C : List (String) | Converts list of conditions into a single string for displaying on the screen by iterating through the parameter C, concatenating together a string ConditionAsString() using a comma and a space as the delimiter. |
| **Return values** | ConditionAsString : String |
|  | |
| GetChallengeMet (public) | | |
| **Parameters** | Pos : Integer | Returns the Met status of a Challenge at the position of Pos in the Challenges list. |
| **Return values** | Boolean |

| **Identifier / Data** | | | **Description** |
| --- | --- | --- | --- |
| GetLockDetails (public) | | | |
| **Parameters** | n/a | Used for displaying a challenge’s current status by iterating through the Challenges protected attribute, concatenating together the output string LockDetails which contains a string version of all the challenges for the lock and whether each has been met or not. | |
| **Return values** | LockDetails: String |
|  |  |
| GetLockSolved (public) | | | |
| **Parameters** | n/a | Returns the status showing if a lock has been solved by iterating through the Challenges protected attribute and returning False if there are any unmet ones, otherwise it returns True. | |
| **Return values** | Boolean |
| GetNumberOfChallenges (public) | | | |
| **Parameters** | n/a | Returns the number of Challenges in the Challenges List (the number of challenges in this lock). | |
| **Return values** | Integer |
| SetChallengeMet (public) | | | |
| **Parameters** | Pos : Integer  Value : Boolean | Uses the SetMet() method in the Challenge class to set the Met attribute of a challenge at the position of Pos in the Challenges list to Met or not Met using the Value parameter. | |
| **Return values** | n/a |

## **Class: Card**

| **Identifier / Data** | | **Description** |
| --- | --- | --- |
| <<constructor>> | | |
| **Parameters** | n/a | Initialises the CardNumber protected attribute using the static attribute (class variable) NextCardNumber. It then increments the static attribute (class variable) NextCardNumber which means that it will be the same and updated for all objects of this class.  Initialises the Score protected attribute to 0. |
| **Return values** | n/a |
|  | |
| GetCardNumber (public) | | |
| **Parameters** | n/a | Returns the value of the protected attribute CardNumber. |
| **Return values** | CardNumber :Integer |
| GetDescription (public) | | |
| **Parameters** | n/a | Returns the protected attribute CardNumber casted as a string. |
| **Return values** | CardNumber: String |
| GetScore (public) | | |
| **Parameters** | n/a | Returns the protected attribute Score. |
| **Return values** | Score : Integer |
| Process (public) | | |
| **Parameters** | Deck : CardCollection  Discard : CardCollection  Hand : CardCollection  Sequence : CardCollection  CurrentLock : Lock  Choice : String  CardChoice : Integer | Base class method for the Process() method in derived classes to override. |
| **Return values** | n/a |

## **Class: ToolCard (inherits from Card)**

| **Identifier / Data** | | **Description** |
| --- | --- | --- |
| <<constructor>> | | |
| **Parameters** | args : List | Initialises the following protected attributes:   * ToolType from parameter args[0] * Kit from parameter args[1]   If there are three elements in args it will initialise CardNumber from parameter args[2]; otherwise it will initialise CardNumber by calling the parent constructor.  Invokes the SetScore() method to assign the correct score in the base class for the ToolType. |
| **Return values** | n/a |
| GetDescription (public) | | |
| **Parameters** | n/a | Overrides the GetDescription() method from the base class to return a concatenated string of the ToolType, a space and the Kit for this ToolCard |
| **Return values** | String |
| SetScore (public) | | |
| **Parameters** | n/a | Assigns the correct Score from the protected attribute ToolType. |
| **Return values** | n/a |

## **Class: DifficultyCard (inherits from Card)**

| **Identifier / Data** | | **Description** |
| --- | --- | --- |
| <<constructor>> | | |
| **Parameters** | args : List | Initialises the protected attribute CardType to ‘Dif’.  If there is one element in args it will initialise CardNumber from parameter args[1]; otherwise it will initialise CardNumber by calling the parent constructor. |
| **Return values** | n/a |
|  | |
| GetDescription (public) <<override>> | | |
| **Parameters** | n/a | Overrides the GetDescription() method from the base class to return the protected attribute CardType. |
| **Return values** | String |
| Process (public) <<override>> | | |
| **Parameters** | Deck : CardCollection  Discard : CardCollection  Hand : CardCollection  Sequence : CardCollection  CurrentLock : Lock  Choice : String  CardChoice : Integer | Overrides the Process() method from the base class to process the user choices from a difficulty card. When the user receives a difficulty card they are asked if they would like to discard a key or 5 cards from the deck.  On choosing the option to discard a key, they are asked to select a key. This method then confirms if the choice parameter is valid. ***Although there are potential logic errors in this check, AQA have confirmed that the code is written as it was intended.***  If the Choice parameter contains the position (it will be converted to an index by subtracting 1 from the position of a ‘key’ ToolCard in the player’s Hand, the card is removed from the Hand and placed in the Discard CardCollection.  If the Choice parameter does not point to a key (either through deliberate user choice or a logic error), 5 cards are removed from the Deck and placed in the Discard CardCollection. |
| **Return values** | n/a |
|  | |

## **Class: CardCollection**

| **Identifier / Data** | | **Description** |
| --- | --- | --- |
| <<constructor>> | | |
| **Parameters** | N : String | Initialises the following protected attributes:   * Name from parameter N * Cards to an empty list |
| **Return values** | n/a |
| GetCardDescriptionAt (public) | | |
| **Parameters** | X : Integer | Returns a string containing the description of the Card at index X in the Cards list by invoking the overridden GetDescription() method in Card. |
| **Return values** | String |
| GetCardNumberAt (public) | | |
| **Parameters** | X : Integer | Returns the CardNumber attribute of a Card at the index X in the Cards list. |
| **Return values** | Integer |
| GetName (public) | | |
| **Parameters** | n/a | Returns the value of the protected attribute Name. |
| **Return values** | Name : String |
| AddCard (public) | | |
| **Parameters** | C (Card) | Appends the value of parameter C to the protected list attribute Cards. |
| **Return values** | n/a |
| CreateLineOfDashes (private) | | | |
| **Parameters** | Size : Integer | Used in formatting a CardCollection display UI.  Returns an appropriately sized LineOfDashes for the number of elements in a CardCollection or fixed at 10 if the CardCollection is greater than that (defined by parameter Size). | |
| **Return values** | LineOfDashes : String |
|  | |
| GetCardDisplay (public) | | | |
| **Parameters** | n/a | Used in formatting a CardCollection display UI. Creates the display output of a CardCollection by concatenating together the collection Name and card descriptions from the protected list attribute Cards. If there are no cards in the list, the collection name and ‘empty’ is returned.  If there are cards in the collection, a list of dashes is created which is either appropriately sized for the number of cards in the collection or is fixed at 10 if the number of cards in the collection is greater than 10. This is to ensure that the display fits correctly in the terminal window.  It then uses indefinite iteration to loop through the Cards list using the GetDescription() method to get a string description of the card at each element and concatenate it with a space and the **|** (pipe) symbol to create a visual ‘line of cards’.  It then creates a second line of dashes to concatenate underneath the ‘line of cards’ and returns the completed output. | |
| **Return values** | CardDisplay : String |
|  | |

| **Identifier / Data** | | **Description** |
| --- | --- | --- |
| GetNumberOfCards (public) | | |
| **Parameters** | n/a | Returns the number of cards in the protected list attribute Cards. |
| **Return values** | Integer |
| RemoveCard (public) | | | |
| **Parameters** | CardNumber : Integer | Returns the card from Cards list at the index CardNumber and removes it from Cards.  If CardNumber is not a valid index, the value of the uninitialised variable CardToGet isreturned. | |
| **Return values** | CardtoGet : Card |
| Shuffle (public) | | | |
| **Parameters** | n/a | Uses definite iteration to perform 10000 movements of cards from one random position to another in the protected list attribute Cards in order to generate a pseudo random shuffle. | |
| **Return values** | n/a |