**Java Journal Template**

**Directions:** Follow the directions for each part of the journal template. Include in your response all the elements listed under the Requirements section. Prompts in the Inspiration section are not required; however, they may help you to fully think through your response.

Remember to review the Touchstone page for entry requirements, examples, and grading specifics.

**Name: Jason Victor**

**Date: 07/16/23**

**Final Replit Program Join Link:**

Complete the following template. Fill out all entries using complete sentences.

PART 1: Defining Your Problem

|  |
| --- |
| **Task**  State the problem you are planning to solve.    **Requirements**   * Describe the problem you are trying to solve. * Describe any input data you expect to use. * Describe what the program will do to solve the problem. * Describe any outputs or results the program will provide.     **Inspiration**  When writing your entry below, ask yourself the following questions:   * Is your problem clearly defined? * Why do you want to solve this particular problem? * What source(s) of data do you believe you will need? Will the user need to supply that data, or will you get it from an external file or another source? * Will you need to interact with the user throughout the program? Will users continually need to enter data in and see something to continue? * What are your expected results or what will be the end product? What will you need to tell a user of your program when it is complete? |
| My goal is to create a strong password generator in order to strengthen security. People need access to a tool that can generates strong and secure passwords. This program prompts users for the desired password length. Also, it will use random characters and predefined rules to generate strong passwords. It will display the generated password and provide users the options to copy it to the clipboard or save it securely.  This program addresses weak passwords and empowers users to protect their information. This program does not need external data sources. It interacts with the user by asking for the desired password length and provides the generated password. The program emphasizes how important it is to use strong passwords. Also, it highlights the significance of password security. |

PART 2: Working Through Specific Examples

|  |
| --- |
| **Task**  Write down clear and specific steps to solve a simple version of your problem you identified in Part 1.    **Requirements**  Complete the three steps below **for at least two distinct examples/scenarios**.   * State any necessary input data for your simplified problem. * Write clear and specific steps in English (not Java) detailing what the program will do to solve the problem. * Describe the specific result of your example/scenario.     **Inspiration**  When writing your entry below, ask yourself the following questions:   * Are there any steps that you don’t fully understand? These are places to spend more time working out the details. Consider adding additional smaller steps in these spots. * Remember that a computer program is very literal. Are there any steps that are unclear? Try giving the steps of your example/scenario to a friend or family member to read through and ask you questions about parts they don’t understand. Rewrite these parts as clearly as you can. * Are there interesting edge cases for your program? Try to start one of your examples/scenarios with input that matches this edge case. How does it change how your program might work? |
| Scenario 1:  Input data: User’s desired password length (using 12 characters)  1. Ask the user to enter the desired length of the new password.  2. Generate a random combination of uppercase and lowercase letters, numbers, and special characters according to the user’s specified length.  3. Make sure that the generated password meets the complexity requirements by including at least one uppercase letter, one lowercase letter, one number, and one special character.  4. Display the generated password.  5. Provide options for the user to either copy the password or save it in a secure place.  6. The result: Generates a strong password with a length of 12 characters, including a combination of uppercase and lowercase letters, numbers, and special characters. The desired password is displayed to the user and can be copied or saved in a secure place.  Scenario 2:  Input data: User’s desired password length (using 8 characters)  1. Ask the user to enter the desired length of the new password.  2. Generate a random combination of uppercase and lowercase letters, numbers, and special characters according to the user’s specified length.  3. Make sure that the generated password meets the complexity requirements by including at least one uppercase letter, one lowercase letter, one number, and one special character.  4. Display the generated password.  5. Provide options for the user to either copy the password or save it in a secure place.  6. The result: Generates a strong password with a length of 8 characters, including a combination of uppercase and lowercase letters, numbers, and special characters. The desired password is displayed to the user and can be copied or saved in a secure place. |

PART 3: Generalizing Into Pseudocode

|  |
| --- |
| **Task**  Write out the general sequence your program will use, including all specific examples/scenarios you provided in Part 2.    **Requirements**   * Write pseudocode for the program in English but refer to Java program elements where they are appropriate. The pseudocode should represent the full functionality of the program, not just a simplified version. Pseudocode is broken down enough that the details of the program are no longer in any paragraph form. One statement per line is ideal.     **Help With Writing Pseudocode**   * Here are a few links that can help you write pseudocode with examples. Remember to check out part 3 of the Example Journal Template Submission if you have not already. Note: everyone will write pseudocode differently. There is no right or wrong way to write it, other than to make sure you write it clearly and in as much detail as you can so that it should be easy to convert to code later. * <https://www.geeksforgeeks.org/how-to-write-a-pseudo-code/> * <https://www.wikihow.com/Write-Pseudocode>     **Inspiration**  When writing your entry below, ask yourself the following questions:   * Do you see common program elements and patterns in your specific examples/scenarios in Part 2, like variables, conditionals, functions, loops, and classes? These should be part of your pseudocode for the general sequence as well. * Are there places where the steps for your examples/scenarios in Part 2 diverged? These may be places where errors may occur later in the project. Make note of them. * When you are finished with your pseudocode, does it make sense, even to a person that does not know Java? Aim for the clearest description of the steps, as this will make it easier to convert into program code later. |
| **Main Class**  Declare constants: UPPERCASE\_LETTERS, LOWERCASE\_LETTERS, NUMBERS, SPECIAL\_CHARACTERS    **Function main()**  Prompt the user to enter the desired password length and assign the returned value to a length variable.  Generate a strong password by calling the generateStrongPassword(length) function and assign the result to the password variable.  Display the generated password by calling the displayPassword(password) function.  Prompt the user for a chosen action (copy or save) by calling the performUserAction(password) function.    **Function getDesiredLength()**  Prompt the user to enter the desired password length.  Return the value of the length variable to the main function.    **Function generateStrongPassword(length)**  Concatenate the UPPERCASE\_LETTERS, LOWERCASE\_LETTERS, NUMBERS, and SPECIAL\_CHARACTERS into a single string called validChars.  Initialize an empty string called password to store the generated password.  Create a for loop that iterates length times.  Within each iteration, generate a random index using the length of validChars.  Append the character at the random index in validChars to the password string.  Return the generated password to the main function.    **Function displayPassword(password)**  Display the message "Generated Password: " followed by the value of the password variable.    **Function performUserAction(password)**  Display a message asking the user to choose an action.  Read the user's input and assign it to the action variable.  If the action value is equal to "copy",  then call the copyToClipboard(password) function.  Else if the action value is equal to "save",  then call the savePassword(password) function.  Else,  call the displayErrorMessage("Invalid action") function.    **Function copyToClipboard(password)**  Display a message showing that the password has been copied to the clipboard.    **Function savePassword(password)**  Display a message showing that the password has been saved successfully.    **Function displayErrorMessage(message)**  Display an error message with the provided message. |

PART 4: Testing Your Program

|  |
| --- |
| **Task**  While writing and testing your program code, describe your tests, record any errors, and state your approach to fixing the errors.    **Requirements**   * For at least one of your test cases, describe how your choices for the test helped you understand whether the program was running correctly or not.   For each error that occurs while writing and testing your code:   * Record the details of the error from Replit. A screenshot or copy-and-paste of the text into the journal entry is acceptable. * Describe what you attempted in order to fix the error. Clearly identify which approach was the one that worked.     **Inspiration**  When writing your entry below, ask yourself the following questions:   * Have you tested edge cases and special cases for the inputs of your program code? Often these unexpected values can cause errors in the operation of your program. * Have you tested opportunities for user error? If a user is asked to provide an input, what happens when they give the wrong type of input, like a letter instead of a number, or vice versa? * Did the outcome look the way you expected? Was it formatted correctly? * Does your output align with the solution to the problem you coded for? |
| The first version of my password generator program that asked the user for a desired password length did not handle the input of a negative number correctly. Please see the screenshot below for this problem:    I added the following try and catch blocks to handle the NegativeArraySizeException.  public static void main(String[] args) {  try {  int length = getDesiredLength();  String password = generateStrongPassword(length);  displayPassword(password);  performUserAction(password);  } catch (InvalidInputException e) {  System.out.println("Invalid input: " + e.getMessage());  } catch (Exception e) {  System.out.println("An error occurred: " + e.getMessage());  }  }    public static int getDesiredLength() throws InvalidInputException {  Scanner input = new Scanner(System.in);  int length;    System.out.print("Enter the desired password length: ");  length = input.nextInt();    if (length <= 0) {  throw new InvalidInputException("Invalid password length. Length must be greater than 0.");  }    return length;  }    public static String generateStrongPassword(int length) throws InvalidInputException {  if (length <= 0) {  throw new InvalidInputException("Invalid password length. Length must be greater than 0.");  }    try {  String validChars = UPPERCASE\_LETTERS + LOWERCASE\_LETTERS + NUMBERS + SPECIAL\_CHARACTERS;  char[] passwordChars = new char[length];  Random random = new Random();    for (int i = 0; i < length; i++) {  int randomIndex = random.nextInt(validChars.length());  passwordChars[i] = validChars.charAt(randomIndex);  }    return new String(passwordChars);  } catch (Exception e) {  throw new InvalidInputException("Failed to generate a strong password.");  }  }  Also, it didn’t continue to ask the user for a valid input after the error exception. I added a while loop to address this issue. Please see the screenshot for this problem:    Here is the while loop:  while (true) {  try {  System.out.print("Enter the desired password length: ");  length = input.nextInt();    if (length <= 0) {  throw new InvalidInputException("Invalid password length. Length must be greater than 0.");  } else {  break;  }  } catch (Exception e) {  System.out.println("Invalid input. Please enter a valid integer value for the password length.");  input.nextLine();  }  }  The program now runs the correct way. Please see the complete new version of the program below.  import java.util.Random;  import java.util.Scanner;    public class Main {  private static final String UPPERCASE\_LETTERS = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  private static final String LOWERCASE\_LETTERS = "abcdefghijklmnopqrstuvwxyz";  private static final String NUMBERS = "0123456789";  private static final String SPECIAL\_CHARACTERS = "!@#$%^&\*()\_-+=<>?";    public static void main(String[] args) {  try {  int length = getDesiredLength();  String password = generateStrongPassword(length);  displayPassword(password);  performUserAction(password);  } catch (InvalidInputException e) {  System.out.println("Invalid input: " + e.getMessage());  } catch (Exception e) {  System.out.println("An error occurred: " + e.getMessage());  }  }    public static int getDesiredLength() throws InvalidInputException {  Scanner input = new Scanner(System.in);  int length;    while (true) {  try {  System.out.print("Enter the desired password length: ");  length = input.nextInt();    if (length <= 0) {  throw new InvalidInputException("Invalid password length. Length must be greater than 0.");  } else {  break;  }  } catch (Exception e) {  System.out.println("Invalid input. Please enter a valid integer value for the password length.");  input.nextLine();  }  }    return length;  }    public static String generateStrongPassword(int length) throws InvalidInputException {  if (length <= 0) {  throw new InvalidInputException("Invalid password length. Length must be greater than 0.");  }    try {  String validChars = UPPERCASE\_LETTERS + LOWERCASE\_LETTERS + NUMBERS + SPECIAL\_CHARACTERS;  char[] passwordChars = new char[length];  Random random = new Random();    for (int i = 0; i < length; i++) {  int randomIndex = random.nextInt(validChars.length());  passwordChars[i] = validChars.charAt(randomIndex);  }    return new String(passwordChars);  } catch (Exception e) {  throw new InvalidInputException("Failed to generate a strong password.");  }  }    public static void displayPassword(String password) {  System.out.println("Generated Password: " + password);  }    public static void performUserAction(String password) {  Scanner input = new Scanner(System.in);  System.out.print("Choose an action (copy/save): ");  String action = input.nextLine();    if (action.equalsIgnoreCase("copy")) {  copyToClipboard(password);  } else if (action.equalsIgnoreCase("save")) {  savePassword(password);  } else {  displayErrorMessage("Invalid action");  }  }    public static void copyToClipboard(String password) {  System.out.println("Password copied to clipboard.");  }    public static void savePassword(String password) {  System.out.println("Password saved successfully.");  }    public static void displayErrorMessage(String message) {  System.out.println("Error: " + message);  }    public static class InvalidInputException extends Exception {  public InvalidInputException(String message) {  super(message);  }  }  } |

PART 5: Commenting Your Program

|  |
| --- |
| **Task**  Submit your full program code, including thorough comments describing what each portion of the program should do when working correctly.    **Requirements**   * The purpose of the program and each of its parts should be clear to a reader that does not know the Java programming language.     **Inspiration**  When writing your entry, you are encouraged to consider the following:   * Is each section or sub-section of your code commented to describe what the code is doing? * Give your code with comments to a friend or family member to review. Add additional comments to spots that confuse them to make it clearer. |
| Here is the complete program code with comments for each section:  import java.util.Random;  import java.util.Scanner;    public class Main {  // Declare constant strings for all of the character sets  private static final String UPPERCASE\_LETTERS = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  private static final String LOWERCASE\_LETTERS = "abcdefghijklmnopqrstuvwxyz";  private static final String NUMBERS = "0123456789";  private static final String SPECIAL\_CHARACTERS = "!@#$%^&\*()\_-+=<>?";    public static void main(String[] args) {  try {  // Get the desired password length from the user  int length = getDesiredLength();  // Generate a strong password of the specified length  String password = generateStrongPassword(length);  // Display the generated password  displayPassword(password);  // Perform the user's chosen action on the password (copy or save)  performUserAction(password);  } catch (InvalidInputException e) {  // Catch and handle any InvalidInputException thrown during user input or password generation  System.out.println("Invalid input: " + e.getMessage());  } catch (Exception e) {  // Catch and handle any other general exceptions that may occur while running this program  System.out.println("An error occurred: " + e.getMessage());  }  }    public static int getDesiredLength() throws InvalidInputException {  Scanner input = new Scanner(System.in);  int length;    while (true) {  try {  // Prompt the user to enter the desired password length  System.out.print("Enter the desired password length: ");  length = input.nextInt();    // Validate the input of the password length  if (length <= 0) {  // Throw an InvalidInputException if the length is zero or a negative number  throw new InvalidInputException("Invalid password length. Length must be greater than 0.");  } else {  // Exit the loop and return the valid password length  break;  }  } catch (Exception e) {  // Catch any input exceptions such as a non-integer inpu and provide an error message  System.out.println("Invalid input. Please enter a valid integer value for the password length.");  input.nextLine(); // Clear the temporary storage of the input  }  }    return length;  }    public static String generateStrongPassword(int length) throws InvalidInputException {  // Validate the input length  if (length <= 0) {  // Throw an InvalidInputException if the length is zero or a negative number  throw new InvalidInputException("Invalid password length. Length must be greater than 0.");  }    try {  String validChars = UPPERCASE\_LETTERS + LOWERCASE\_LETTERS + NUMBERS + SPECIAL\_CHARACTERS;  char[] passwordChars = new char[length];  Random random = new Random();    // Generate each character of the password randomly from any of the valid character sets  for (int i = 0; i < length; i++) {  int randomIndex = random.nextInt(validChars.length());  passwordChars[i] = validChars.charAt(randomIndex);  }    return new String(passwordChars);  } catch (Exception e) {  // Catch any exceptions during password generation and throw an InvalidInputException  throw new InvalidInputException("Failed to generate a strong password.");  }  }    public static void displayPassword(String password) {  // Display the generated password  System.out.println("Generated Password: " + password);  }    public static void performUserAction(String password) {  Scanner input = new Scanner(System.in);  System.out.print("Choose an action (copy/save): ");  String action = input.nextLine();    // Perform the user's chosen action on the password  if (action.equalsIgnoreCase("copy")) {  copyToClipboard(password);  } else if (action.equalsIgnoreCase("save")) {  savePassword(password);  } else {  // Display an error message for an invalid action  displayErrorMessage("Invalid action");  }  }    public static void copyToClipboard(String password) {  // Display copying the password to the clipboard  System.out.println("Password copied to clipboard.");  }    public static void savePassword(String password) {  // Display saving the password  System.out.println("Password saved successfully.");  }    public static void displayErrorMessage(String message) {  // Display an error message  System.out.println("Error: " + message);  }    public static class InvalidInputException extends Exception {  // Custom exception class to represent an invalid input  public InvalidInputException(String message) {  super(message);  }  }  } |

PART 6: Your Completed Program

|  |
| --- |
| **Task**  Provide the Replit link to your full program code.    **Requirements**   * The program must work correctly with all the comments included in the program.     **Inspiration**   * Check before submitting your Touchstone that your final version of the program is running successfully. |
| https://replit.com/@JasonVictor3/PasswordGenerator-1 |