UML DIAGRAMS:

|  |
| --- |
| Player Class: |
| Random r = new random() – gets a new instance of the random class.  String name – holds the name of the player.  int health = 50 – holds the health of the player.  String[] offensive\_spells = new String[] – holds all the attack spells.  String[] defensive\_spells = new String[] – holds all the spells that defend. |
| void printHealthReduction(int amountDecreased) – prints the reduction of health from a spell based on amountDecreased.  void setHealth(int value) – takes in an integer value and sets the players health based on that.  int getHealth() -- returns the character’s health.  void setName(String name) – sets Character name to name.    String getName() – returns Character’s name.  int roll() -returns a random number between 1-6.  String getOSpell(int playerRoll) – returns the spell used by the offensive player.    String getDSpell(int playerRoll) – returns the spell used by the defensive player.    void set\_Health\_OffensePlayer(int playerRoll, int opponentRoll)   * sets health based on outcome of play rolls. You will use the roll results as parameters .     void set\_Health\_DefensePlayer(int playerRoll, int opponentRoll)   * sets health based on outcome of play rolls. You will use the roll results as parameters. |

|  |
| --- |
| Game Class: |
| Some variables are private and static meaning they can be only used in this class, and that they do not required an instance of the class to be used!  private static int player1Roll = 0 – holds the dice roll of the player.  private static int player2Roll = 0 – holds the dice roll of the player.  private static boolean currentPlayer = true – controls which player is attacker, if true player 1 is attacker, if false player 2 is attacker.  private static Player player1 = new Player() – gets an instance of the player class.  private static Player player2 = new Player() – gets an instance of the player class. |
| void blankLine() – prints a space when called.  void printMenu() – prints a menu to the console for the user.  void managePlayers() – manages the players using the player class and currentPlayer boolean.  void printHealhUpdate() – prints the current health of the player.  void main(String[] args) – runs the main program that calls all methods above and the player class to simulate the Harry Potter game.  Variables of main:  boolean gameLoop = false – manages if the game is running.  boolean programLoop = true – manages if the program is running.  String userInput – holds the input from the user.  Scanner input – new Scanner(System.in) – gets a new instance of the scanner class to be used to get input from the user. |

NAME: Michael Bradshaw TOTAL: / 40

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **CATEGORY** | **CRITERIA** | **< LEVEL 1**  **0 – 49%** | **LEVEL 1 50 – 59%** | **LEVEL 2 60 – 69%** | **LEVEL 3 70 – 79%** | **LEVEL 4 80 – 100%** | **MARK** |
| **Knowledge and**  **Understanding** | Demonstrates an understanding of how to create a Character & Dice class and how to use the methods included | • Demonstrates  little or no  understanding of how to create a Character & Dice class  **0-4.9** | • Demonstrates limited understanding of how to create a Character &  Dice class  **5.0-5.9** | • Demonstrates some understanding of how to create a Character &  Dice class  **6.0-6.9** | • Demonstrates considerable understanding of how to create a Character &  Dice class  **7.0-7.9** | • Demonstrates thorough understanding of how to create a Character & Dice class  **8.0-10** | **/10** |
| **Thinking** | The program meets all the specifications required        Validates program to ensure the program produces correct results | * Program   meets little or none of the required  specifications       * Validates program with   little or no  success    **0-4.9** | * Program meets a limited number of the required specifications      * Validates program with limited   success    **5.0-5.9** | * Program meets some of the required   specifications       * Validates program with some   success    **6.0-6.9** | * Program meets most of the required   specifications       * Validates program with considerable   success    **7.0-7.9** | * Program   meets all of the required  specifications         * Validates program with   great success      **8.0-10** | **/10** |
| **Communication** | Provides internal documentation that clearly explains the methods and the program logic | • Documents  program logic  with little or no success      **0-4.9** | • Documents  program logic  with limited  success      **5.0-5.9** | • Documents  program logic with some  success      **6.0-6.9** | • Documents  program logic  with considerable success    **7.0-7.9** | • Documents  program logic with great success      **8.0-10** | **/10** |
| **Application** | Effectively applies programming  skills and knowledge of Java to create a program using a Character & Dice class and the String class | • Applies programming  knowledge  and skills with  little or no  success    **0-4.9** | • Applies programming  knowledge  and skills with limited  success    **5.0-5.9** | • Applies programming  knowledge  and skills with some  success    **6.0-6.9** | • Applies programming  knowledge  and skills with considerable  success    **7.0-7.9** | • Applies programming  knowledge  and skills with great success      **8.0-10** | **/10** |

**CURRICULUM EXPECTATIONS THAT ARE COVERED IN THIS ASSIGNMENT:**

|  |  |
| --- | --- |
| A3.1 | Demonstrate the ability to use existing subprograms within computer programs. |
| A3.2 | Write subprograms that use parameter passing and appropriatevariable scope to performtasks within programs. |
| A4.1 | Demonstrate the ability to identify and correct syntax, logic and run-time errors in computer programs. |
| A4.2 | Use workplace and professional conventions correctly to write programs and internal documentation. |
| A4.5 | Demonstrate the ability to validate a program using test cases. |
| B2.1 | Design programs from a program template or skeleton. |
| B2.3 | Apply the principle of modularity to design reusable code *(e.g., subprograms, classes)* in computer programs. |
| B2.4 | Represent the structure and components of a program using industry-standard programming tools. |