PART A – Code Comprehension

1) Main method creates an instance of GameCharacters.
Usin this we call menuSystem function.
Code:

```
public static void main(String[] args) {

                    // Create a new object of myself
                    GameCharacters c= new GameCharacters();
                    // Provide a menu to the console and act on the user response
                    c.menuSystem();
          }
```

The following line is used to read user input inside menuSystem,

```
int menuSelection = myScanner.nextInt();  // read selection from user
```

2) '\t' is tab seperator used to put tabular space between two strings.

3) NCHARACTERS = 3 specifies the number of characters.
NATTRIBUTES = 8 specifies the number of attributes.
Code:

```
          static final int NCHARACTERS = 3;
          static final int NATTRIBUTES = 8;
          int[][] tableOfCharacters = new int[NCHARACTERS][NATTRIBUTES];
```

4) Message: Invalid option. Must be between 1 and 6 or 99 for exit.

Code:

```
if (menuSelection < 1 || menuSelection > 6)
          System.out.println ("Invalid option. Must be between 1 and 6 or 99 for exit.");
```

5) Output:

```
Printing Formatted Details of All Characters
==========================================
No      Code            Student ID      Eyes    Legs    BirthYr Month           Day     Health
        Power
0       0       1       ??      0       0       0       0       Poor Health     0
1       0       1       ??      0       0       0       0       Poor Health     0
2       0       1       ??      0       0       0       0       Poor Health     0
    --- End of List ---
```

displayAllCharactersFormatted() function is responsible for this

6) Code:

```
          public int createCharacterCode(int x) {

                    if (almostCharacterCode == 0) {
                              almostCharacterCode = studentID;
                              almostCharacterCode = almostCharacterCode * FACTOR;
```

```
                almostCharacterCode = almostCharacterCode + LAST_TWO_CHARS;
                almostCharacterCode = almostCharacterCode - CHARS_FIVE_AND_SIX;
            }
            // With our encoded StudentID, add the passed value (in this case, the row value)
            characterCode = almostCharacterCode + x;
            return characterCode;

        }
```

if almostCharacterCode is 0, we initialize its value using studentId. We encode it by multiplying it by FACTOR, adding LAST_TWO_CHARS and then subracting CHARS_FIVE_AND_SIX.
A For each characters we use almostCharacterCode to create character's code.

7) Output:

```
Line  _code  _row  decodedCharCode
314 | 123456768 | 1 | not defined
315 | 123456768 | 1 | 123456768
316 | 123456768 | 1 | 123456767
317 | 123456768 | 1 | 123456845
318 | 123456768 | 1 | 123456780
319 | 123456768 | 1 | 12345678
```

```
Printing Formatted Details of All Characters
==========================================
No    Code          Student ID    Eyes    Legs    BirthYr Month       Day    Health
      Power
0     123456767     12345678      Blue    1       1911    1       1      Fair Health    100
1     123456768     12345678      Green   2       1922    2       2      Med Health     200
2     123456769     12345678      Hazel   3       1933    3       3      Good Health    300
   --- End of List ---
```

8) Line no 212 causes the issue.
almostCharacterCode = almostCharacterCode * FACTOR;
Here, almostCharacterCode * FACTOR ranges beyond integer maximum limit hence becomes negative.

9) "/" will divide the FACTOR instead of multiplying.
almostCharacterCode is an int datatype.
It should be changed to float

10) On removing break from line 297, the program goes to next block and execute it. Then it encounters break of the next block and comes out of switch case block.

An exception was thrown:

```
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 123456767
        at GameCharacters.displayCharacterDetails(GameCharacters.java:299)
        at GameCharacters.displayAllCharactersFormatted(GameCharacters.java:281)
        at GameCharacters.menuSystem(GameCharacters.java:127)
        at GameCharacters.main(GameCharacters.java:109)
```

Because EYES array goes out of bound.

11) Executes 3 times. Once for eachcharacter. We can use 'x' also for this purpose.

12) This flag is set to true when user enters a valid powerand operation is performed. So that user can come out of that operation

PART B – Programming Requirements

GameCharacters.java

```
//package assignment1;

/* Author: Tom Keogh
 * Creation Date: 22 November 2015
 * Last Modified: 27 November 2015
 * This is starter code for Assignment 1 ITECH1000 - Programming 1 - for Summer semester 2015
(201527) at Federation University.
 * This class allows for the entry of a pre-defined number of characters - NCHARACTERS - that
might be used, for example,
 * in a game and then allows for some reporting and functionality on those characters. At this stage
we use this
 * GameCharacters class for collecting information about multiple (but a predefined number)
objects (game characters) and
 * providing the interface and application functionality attached to these game characters. As you
learn more about object oriented
 * concepts you will realise that some of this should be separated into other classes but this format
will do as an
 * introduction. We've also left out any notion of visibility or encapsulation other than to make all
methods public and accept
 * the default visibility on variables. Finally, some inconsistency in coding style is intentional as it
is written in a way
 * that shows different ways of coding a solution including constant or parameter driven methods
and constructs and variations
 * on these. For example, some methods use parameters whilst others do not but could. Some code
is written in a way so that only
 * constants need changing whilst other code uses literals and less dynamic means.
 * Students are encouraged to study this code and understand it with a view to consolidating their
understanding of sequencing,
 * selection and repetition. Whilst this is an individual assignment, a first useful step might be to
discuss the starter code with
 * your peers and lecturer.
 */

import java.util.Scanner;

public class GameCharacters {

        // constants
        // constant for exiting system
        static final int EXITSYSTEM = 99;
	// provide constants for array size
	static final int NCHARACTERS = 3;
	static final int NATTRIBUTES = 8;

	// place to store characters, one row of attributes for each character input
	int[][] tableOfCharacters = new int[NCHARACTERS][NATTRIBUTES];

	// provide constants for ranges of acceptable values
	static final int MAX_EYES=3, MIN_EYES=0,
					MAX_LEGS=8, MIN_LEGS=0,
					MAX_YEAR=2100, MIN_YEAR=1900,
					MAX_MONTH=12, MIN_MONTH=1,
					MAX_DAY=31, MIN_DAY=1,
					MAX_HEALTH=4, MIN_HEALTH=0,
					MAX_POWER=20000, MIN_POWER=0;

	// constants for messages and headings
	static final String EYES[]   = {"??", "Blue", "Green","Hazel"},
						HEALTH[] = {"Poor Health", "Fair
Health",
								"Med Health",
"Good Health",
								"Excel. Health"},
						ATTR_HEADINGS = "No    \tCode
\t\tStudent ID \tEyes \tLegs \tBirthYr Month \tDay \tHealth \t\tPower",
						END_REPORT = "     --- End of List ---
";

	// some text for communicating with user
	static final String INTRO_EYES  = "What colour eyes will the character have? ",
						INTRO_LEGS  = "How many legs?
Must be between ",
						INTRO_YEAR  = "Year of Birth in
CCYY? Must be between ",
						INTRO_MONTH = "Month of Birth?
Must be between ",
						INTRO_DAY   = "Day of Birth? Must
be between ",
						INTRO_HEALTH= "What level of
health does the character have? ",
						INTRO_POWER = "What level of
power does the character have? Must be between ";

	// position of attributes in array row
	static final int CODE=0, EYE_COLOUR=1, LEGS=2, YEAR=3,
					MONTH=4, DAY=5, HEALTHINESS=6,
POWER=7;

	// powerful items and their value that can be added to a character's power
	static final String POWERITEM[] = {"Weapon", "Crystal", "Food", "Water","Fire","Ice"};
// Adding two more power types
```

```
	static final int ITEMVAL[] = {5000, 8000, 3000, 5000, 1000,2000};   //Adding their values

	// Maximum and minimum studentID values
	static final int MAXSTUDENTID = 99999999,
	                                MINSTUDENTID = 10000000;

	// some constants to modify your studentID and create a character code
	static final int FACTOR=10, LAST_TWO_CHARS=65, CHARS_FIVE_AND_SIX=78;

	// instance variables
	// At construction, Java initialises instance primitives and instance
	// object references (but not local/method variables) to defaults when
	// variables have not been initialised e.g. 0 for int and null for object
	// references. Good practice though to initialise yourself since you
	// control the value initialised to.

	// scanner to enable input from user
	Scanner myScanner = new Scanner(System.in);

	// place for storing entered studentID
	int studentID = 0;
	// place for storing encoded studentID
	int almostCharacterCode = 0;
	// place for storing character code
	int characterCode = 0;
	// has data been entered?
	boolean tableOfCharactersFilled = false;

	// Default  or no-argument Constructor:
	// Java implicitly provides default constructor if no other constructor
	// is declared, but it's good practice to explicitly declare default constructor
	public GameCharacters() {

	}

	public static void main(String[] args) {

		// Create a new object of myself
		GameCharacters c= new GameCharacters();
		// Provide a menu to the console and act on the user response
		c.menuSystem();
	}

	public void menuSystem () {

		printMenu();  // display menu options to screen


		int menuSelection = myScanner.nextInt();  // read selection from user
		while (menuSelection != EXITSYSTEM) {  // so long as user has not selected exit
value, do this loop
```

```
            if (!tableOfCharactersFilled) {
                if (menuSelection == 1) createCharacters();
                else System.out.println ("You need to create your characters before
choosing any other menu options");
            }
            else {
                if (menuSelection == 1) System.out.println ("You've already created
your characters. Choose another option");
                else if (menuSelection == 2) displayAllCharactersRaw();
                else if (menuSelection == 3) displayAllCharactersFormatted();
                else if (menuSelection == 4) displayOneCharacter();
                else if (menuSelection == 5) displayTotalLegsAndPower();
                else if (menuSelection == 6) addPowerToCharacter();
            }
            if (menuSelection < 1 || menuSelection > 6)
                System.out.println ("Invalid option. Must be between 1 and 6 or 99 for
exit.");

            printMenu();  // print menu again for repeat loop
            menuSelection = myScanner.nextInt();  // read next selection from user
        }
        // We have menuSelection == EXITSYSTEM so send message and close Scanner
        System.out.println ("Exiting system");
        myScanner.close();
    }

    public  void printMenu () {
        // Display the options. Could have been more sophisticated here and looped through
        // two arrays of constants - one with options (int) and one with corresponding
        // text (String) but thought it might be better to show another way
        System.out.println("\nMenu");
        System.out.println("====");
        System.out.println("Choose options to create, display or enhance your characters");
        System.out.println("Option 1: Create your characters");
        System.out.println("Option 2: Display entered information of all characters");
        System.out.println("Option 3: Display formatted information of all characters");
        System.out.println("Option 4: Display a particular character");
        System.out.println("Option 5: Display total legs and power for all characters");
        System.out.println("Option 6: Empower your character");
        System.out.println("Option 99: Quit");
    }

    public  void createCharacters () {
        // Ask user to key in values for characters. All details of all characters are entered at
the one time.

        // Firstly get the student ID and store it
        // We should probably ask for a String and do some validation but the
        // following is a crude but sufficient way at this stage of your learning
        System.out.println("What is your student id? Must be between " +
MINSTUDENTID + " and " + MAXSTUDENTID);
        while ( myScanner.hasNext()) {
            studentID = myScanner.nextInt();
            if(studentID > MINSTUDENTID && studentID < MAXSTUDENTID){
                    break;
              }
              System.out.println("What is your student id? Must be between " +
MINSTUDENTID + " and " + MAXSTUDENTID);

      }
      /*
              while (studentID < MINSTUDENTID || studentID > MAXSTUDENTID) {
                    System.out.println("What is your student id? Must be between " +
MINSTUDENTID + " and " + MAXSTUDENTID);
                    studentID = myScanner.nextInt();
              }
              */
              // Now start adding character data
              // for all the rows
              for (int i=0; i<NCHARACTERS; i++){
                    System.out.println("Character " + i);
                    // Now for every row take studentID (stored globally)
                    // and row number and create a character code
                    tableOfCharacters[i][CODE] = createCharacterCode(i);

                    // Add the other attributes

                    // Create an empty array when we have no array information to send because
we must satisfy the method signature
                    String [] dummyArray = {};
                    // Add eye colour value
                    tableOfCharacters[i][EYE_COLOUR] =
obtainCharacterAttribute(INTRO_EYES,     EYES,MIN_EYES, MAX_EYES);
                    // Add number of legs
                    tableOfCharacters[i][LEGS] =
obtainCharacterAttribute(INTRO_LEGS,dummyArray,MIN_LEGS,MAX_LEGS);
                    // Add year of birth
                    tableOfCharacters[i][YEAR] =
obtainCharacterAttribute(INTRO_YEAR,dummyArray,MIN_YEAR,MAX_YEAR);
                    //Add month of birth
                    tableOfCharacters[i][MONTH]         =
obtainCharacterAttribute(INTRO_MONTH,dummyArray,MIN_MONTH,MAX_MONTH);
                    //Add day of birth
                    tableOfCharacters[i][DAY] =
obtainCharacterAttribute(INTRO_DAY,dummyArray,MIN_DAY,MAX_DAY);
                    //Add initial character health
                    tableOfCharacters[i][HEALTHINESS]=
obtainCharacterAttribute(INTRO_HEALTH,HEALTH,MIN_HEALTH,MAX_HEALTH);
                    //Add initial character power
                    tableOfCharacters[i][POWER]=
obtainCharacterAttribute(INTRO_POWER,dummyArray,MIN_POWER,MAX_POWER);

              }
              // character array is now filled
```

```
		tableOfCharactersFilled = true;
	}

	public int createCharacterCode(int x) {
		// This is a very primitive method to encode the StudentID, add the
		// row number and create a character code.
		// We should be more sophisticated by using encoding mechanisms
		// and selecting the values to use from the number itself but at
		// this stage this is sufficient. The values and operations used
		// ensures that the code created is below the largest int storable
		// value - 2^31

		// First time through, encode the studentID
		if (almostCharacterCode == 0) {
			almostCharacterCode = studentID;
			almostCharacterCode = almostCharacterCode * FACTOR;
			almostCharacterCode = almostCharacterCode + LAST_TWO_CHARS;
			almostCharacterCode = almostCharacterCode - CHARS_FIVE_AND_SIX;
		}
		// With our encoded StudentID, add the passed value (in this case, the row value)
		characterCode = almostCharacterCode + x;
		return characterCode;

	}

	// Have a standard dialogue processor to obtain attribute data
	public int obtainCharacterAttribute(String _introText, String[] _typesText, int _min, int
_max) {
		String dialogueText = _introText;
		if (_typesText.length == 0) {
			// if we have no array it's a min and max situation
			// i.e. "attribute must be between " _min + " and " + _max
			dialogueText = dialogueText + _min + " and " + _max;
		}
		else {
			for (int k=0; k<_typesText.length; k++) {
				dialogueText = dialogueText + _typesText[k] + "=" + k;
				// add a comma at the end of each except the last
				if (k!=_typesText.length-1)
				dialogueText = dialogueText + ",";
			}
		}

		// Now initialise what is being asked for to an invalid value
		int returnedValue = _max + 1; // ensures that the while loop below will happen at
least once
		// Continue asking until a valid value is obtained

		do{
			System.out.println(dialogueText);
			returnedValue = myScanner.nextInt();
		}while(returnedValue < _min || returnedValue > _max);
		/*
		while (returnedValue < _min ||
			returnedValue > _max)	{
			System.out.println(dialogueText);
			returnedValue = myScanner.nextInt();
		}
		*/
		return returnedValue;
	}

	public void displayAllCharactersRaw () {
		// Iterate through the 2 dimensional array and display what is there
		// For all the rows

		System.out.println("Printing Entered Details of Characters");
		System.out.println("=====================================");
		String rawDetails; // This String will be used to build up the final string for the
output statement for each row so remember initialisation

		//tableOfCharacters = new int[NCHARACTERS][NATTRIBUTES];
		rawDetails = "";
		for(int i = 0 ; i < NCHARACTERS;i++){
			for(int j = 0 ; j < NATTRIBUTES;j++){
				rawDetails = rawDetails + tableOfCharacters[i][j] + "\t";

			}
			rawDetails = rawDetails + "\n";

		}

		System.out.println(rawDetails);









		System.out.println(END_REPORT);
	}

	public  void displayAllCharactersFormatted () {
		// This will go through the data array and get each character's details and print them

		System.out.println("Printing Formatted Details of All Characters");
		System.out.println("=========================================");
		System.out.println (ATTR_HEADINGS);
		// for each row of the array, display the details

		for (int i=0; i<NCHARACTERS; i++){
				displayCharacterDetails(i);
			}

			System.out.println(END_REPORT);
		}

		public void displayCharacterDetails(int _a) {
			String charAttrs = "";
			// Show the character number
			charAttrs = charAttrs + _a;

			for (int j=0; j<NATTRIBUTES; j++) {

				if (j==CODE){
					charAttrs = charAttrs + " \t" + tableOfCharacters[_a][j];
					charAttrs = charAttrs + " \t" +
decodeCharacterCode(tableOfCharacters[_a][CODE], _a);


				}
				else if (j==EYE_COLOUR){

					charAttrs = charAttrs + " \t" + EYES[tableOfCharacters[_a][j]]  + " ";

				}
				else if (j==HEALTHINESS){
					charAttrs = charAttrs + " \t" + HEALTH[tableOfCharacters[_a][j]]  +
" ";

				}
				else if ((j==LEGS)||(j==YEAR)||(j==MONTH)||(j==DAY)||(j==POWER)){
					charAttrs = charAttrs + " \t" + tableOfCharacters[_a][j]  + " ";
				}
				else{
					System.out.println("Something odd about character " + _a + " in
column " + j);
				}

				/*
				switch (j) {
					case CODE: charAttrs = charAttrs + " \t" + tableOfCharacters[_a][j];
							charAttrs = charAttrs + " \t" +
decodeCharacterCode(tableOfCharacters[_a][CODE], _a);
							break;
					// for eye_colour and healthiness, write out the String, not the number
					case EYE_COLOUR: charAttrs = charAttrs + " \t" +
EYES[tableOfCharacters[_a][j]]  + " ";
							break;
					case HEALTHINESS: charAttrs = charAttrs + " \t" +
HEALTH[tableOfCharacters[_a][j]]  + " ";
							break;
						// for all others (it's good practice to specify), just take the int
value
				case LEGS: case YEAR: case MONTH: case DAY: case POWER:
							charAttrs = charAttrs + " \t" +
tableOfCharacters[_a][j] + " ";
						break;
				default: System.out.println("Something odd about character " + _a + "
in column " + j);
			}
			*/
		}
		// Print a row of character information
		System.out.println (charAttrs);
	}

	public int decodeCharacterCode(int _code, int _row) {
		int decodedCharCode = _code;
		decodedCharCode= decodedCharCode - _row;
		decodedCharCode = decodedCharCode + CHARS_FIVE_AND_SIX;
		decodedCharCode = decodedCharCode - LAST_TWO_CHARS;
		decodedCharCode = decodedCharCode / FACTOR;
		return decodedCharCode;
	}

	public  void displayOneCharacter () {
		// This will ask for a particular character in the position of the array and get each
character's details and print them
		// For simplicity we know we either have three characters or none

		int choice = selectOneCharacter();
		System.out.println("Printing List of One Character");
		System.out.println("==============================");
		System.out.println (ATTR_HEADINGS);
		displayCharacterDetails(choice);
		System.out.println(END_REPORT);

	}

	public  int selectOneCharacter () {
		int choice = -1;
		int maxIndex = NCHARACTERS-1;
		// For simplicity, we don't provide an exit; a choice must be made
		while (choice < 0 || choice > maxIndex ) {
			System.out.println("Please select your character. Must be between 0 and " +
maxIndex);
			choice = myScanner.nextInt();
		}
		return choice;
	}

	public  void addPowerToCharacter () {
		// Let user select the character
		int choice = selectOneCharacter();
		// show the character on the console
		displayCharacterDetails(choice);
		// String to contain combination of power description, power value and option
number
		String powerText = "";
		// To allow the user to finish after validly adding some power
		boolean finished = false;
		// To allow the user to exit when not wanting to enter a value
		int exit = -1;
		// the user selection
		int value = 0;

		while (!finished && value !=exit ) {
			// Ask user for points but point out that you can't go over the max
			System.out.println("How many power points do you want to give to your
character? ");
			System.out.println("Your character can't have more than the maximum power
of " + MAX_POWER);
			// only need to do this the first time
			// build String containing combination of power description, power value and
option number
			if (powerText.length() == 0) {
				for (int k=0; k< ITEMVAL.length; k++) {
					powerText = powerText + POWERITEM[k] + " for " +
ITEMVAL[k] + "=" + k;
					// add a comma at the end of each except the last
					if (k!=ITEMVAL.length-1)
					powerText = powerText + ",";
				}
			// add option to exit
			powerText = powerText + ", -1 to exit.";
			}
			System.out.println(powerText);
			// get the user selection
			value = myScanner.nextInt();
			// if the user hasn't asked to exit
			if (value != exit) {
				// if a valid index of a power value has been chosen
				if (value >= 0 && value <ITEMVAL.length) {
					// get the sum of the original power value for the chosen
character and add the power value
					// associated with the index chosen
					int newPower = ITEMVAL[value] +
tableOfCharacters[choice][POWER];
					// if the new total power value is okay i.e. below the max
					if (newPower <= MAX_POWER) {
						// update the character power value in the table with the
new total power value
						tableOfCharacters[choice][POWER] = newPower;
						// Display a message that update has occurred
						System.out.println ("You've updated the power of
character " + choice + " to " + newPower);
```

```
                                                // and you've finished
                                                finished = true;
                                        }
                                        else {
                                                // You've exceeded the power, try again
                                                System.out.println ("You've exceeded total power for
the character. Try again.");
                                        }
                                }
                        }
                }
        }

        public void displayTotalLegsAndPower() {
                // Iterate through the 2 dimensional array and display the total of
                // legs and the total of power for all characters
                int totalLegs = 0;
                int totalPower = 0;
                System.out.println ("Total Legs and Power for all Characters");
                System.out.println ("=======================================");
                // For all the rows
                int i=0;
                while(i<NCHARACTERS){
                        totalLegs = totalLegs + tableOfCharacters[i][LEGS];
                        totalPower = totalPower + tableOfCharacters[i][POWER];
                        i++;
                }
                /*
                for (int i=0; i<NCHARACTERS; i++) {
                        totalLegs = totalLegs + tableOfCharacters[i][LEGS];
                        totalPower = totalPower + tableOfCharacters[i][POWER];
                }
                */
                // Display the totals
                System.out.println("Total number of legs:  " + totalLegs);
                System.out.println("Total amount of power: " + totalPower);
                System.out.println(END_REPORT);
        }
}
```

PART C – Testing

Here is a sample output for the following program:

```
Menu
====
Choose options to create, display or enhance your characters
Option 1: Create your characters
Option 2: Display entered information of all characters
Option 3: Display formatted information of all characters
Option 4: Display a particular character
Option 5: Display total legs and power for all characters
Option 6: Empower your character
Option 99: Quit
2
You need to create your characters before choosing any other menu options

Menu
====
Choose options to create, display or enhance your characters
Option 1: Create your characters
Option 2: Display entered information of all characters
Option 3: Display formatted information of all characters
Option 4: Display a particular character
Option 5: Display total legs and power for all characters
Option 6: Empower your character
Option 99: Quit
1
What is your student id? Must be between 10000000 and 99999999
12345678
Character 0
What colour eyes will the character have? ??=0,Blue=1,Green=2,Hazel=3
1
How many legs? Must be between 0 and 8
2
Year of Birth in CCYY? Must be between 1900 and 2100
1911
Month of Birth? Must be between 1 and 12
1
Day of Birth? Must be between 1 and 31
1
What level of health does the character have? Poor Health=0,Fair Health=1,Med Health=2,Good Health=3,Excel.
Health=4
1
What level of power does the character have? Must be between 0 and 20000
1000
Character 1
What colour eyes will the character have? ??=0,Blue=1,Green=2,Hazel=3
2
How many legs? Must be between 0 and 8
2
Year of Birth in CCYY? Must be between 1900 and 2100
1922
Month of Birth? Must be between 1 and 12
2
Day of Birth? Must be between 1 and 31
2
What level of health does the character have? Poor Health=0,Fair Health=1,Med Health=2,Good Health=3,Excel.
Health=4
2
What level of power does the character have? Must be between 0 and 20000
2000
Character 2
What colour eyes will the character have? ??=0,Blue=1,Green=2,Hazel=3
3
How many legs? Must be between 0 and 8
4
Year of Birth in CCYY? Must be between 1900 and 2100
1933
Month of Birth? Must be between 1 and 12
3
```

```
Day of Birth? Must be between 1 and 31
3
What level of health does the character have? Poor Health=0,Fair Health=1,Med Health=2,Good Health=3,Excel.
Health=4
3
What level of power does the character have? Must be between 0 and 20000
3000

Menu
====
Choose options to create, display or enhance your characters
Option 1: Create your characters
Option 2: Display entered information of all characters
Option 3: Display formatted information of all characters
Option 4: Display a particular character
Option 5: Display total legs and power for all characters
Option 6: Empower your character
Option 99: Quit
2
Printing Entered Details of Characters
==============================
123456767 1 2 1911 1 1 1 1000
123456768 2 2 1922 2 2 2 2000
123456769 3 4 1933 3 3 3 3000

     --- End of List ---

Menu
====
Choose options to create, display or enhance your characters
Option 1: Create your characters
Option 2: Display entered information of all characters
Option 3: Display formatted information of all characters
Option 4: Display a particular character
Option 5: Display total legs and power for all characters
Option 6: Empower your character
Option 99: Quit
3
Printing Formatted Details of All Characters
==============================
No     Code  Student ID  Eyes  Legs  BirthYr Month  Day  Health  Power
0  123456767  12345678  Blue   2   1911   1   1   Fair Health   1000
1  123456768  12345678  Green   2   1922   2   2   Med Health   2000
2  123456769  12345678  Hazel   4   1933   3   3   Good Health   3000
     --- End of List ---

Menu
====
Choose options to create, display or enhance your characters
Option 1: Create your characters
Option 2: Display entered information of all characters
Option 3: Display formatted information of all characters
Option 4: Display a particular character
Option 5: Display total legs and power for all characters
Option 6: Empower your character
Option 99: Quit
4
Please select your character. Must be between 0 and 2
2
Printing List of One Character
```

```
==============================
No     Code  Student ID  Eyes  Legs  BirthYr Month  Day  Health  Power
2  123456769  12345678  Hazel   4   1933   3   3   Good Health   3000
     --- End of List ---

Menu
====
Choose options to create, display or enhance your characters
Option 1: Create your characters
Option 2: Display entered information of all characters
Option 3: Display formatted information of all characters
Option 4: Display a particular character
Option 5: Display total legs and power for all characters
Option 6: Empower your character
Option 99: Quit
5
Total Legs and Power for all Characters
==============================
Total number of legs:  8
Total amount of power: 6000
     --- End of List ---

Menu
====
Choose options to create, display or enhance your characters
Option 1: Create your characters
Option 2: Display entered information of all characters
Option 3: Display formatted information of all characters
Option 4: Display a particular character
Option 5: Display total legs and power for all characters
Option 6: Empower your character
Option 99: Quit
6
Please select your character. Must be between 0 and 2
0
0  123456767  12345678  Blue   2   1911   1   1   Fair Health   1000
How many power points do you want to give to your character?
Your character can't have more than the maximum power of 20000
Weapon for 5000=0,Crystal for 8000=1,Food for 3000=2,Water for 5000=3,Fire for 1000=4,Ice for 2000=5, -1 to
exit.
3
You've updated the power of character 0 to 6000

Menu
====
Choose options to create, display or enhance your characters
Option 1: Create your characters
Option 2: Display entered information of all characters
Option 3: Display formatted information of all characters
Option 4: Display a particular character
Option 5: Display total legs and power for all characters
Option 6: Empower your character
Option 99: Quit
```