Concordia University
COMP 248 – Winter 2021
Assignment 4

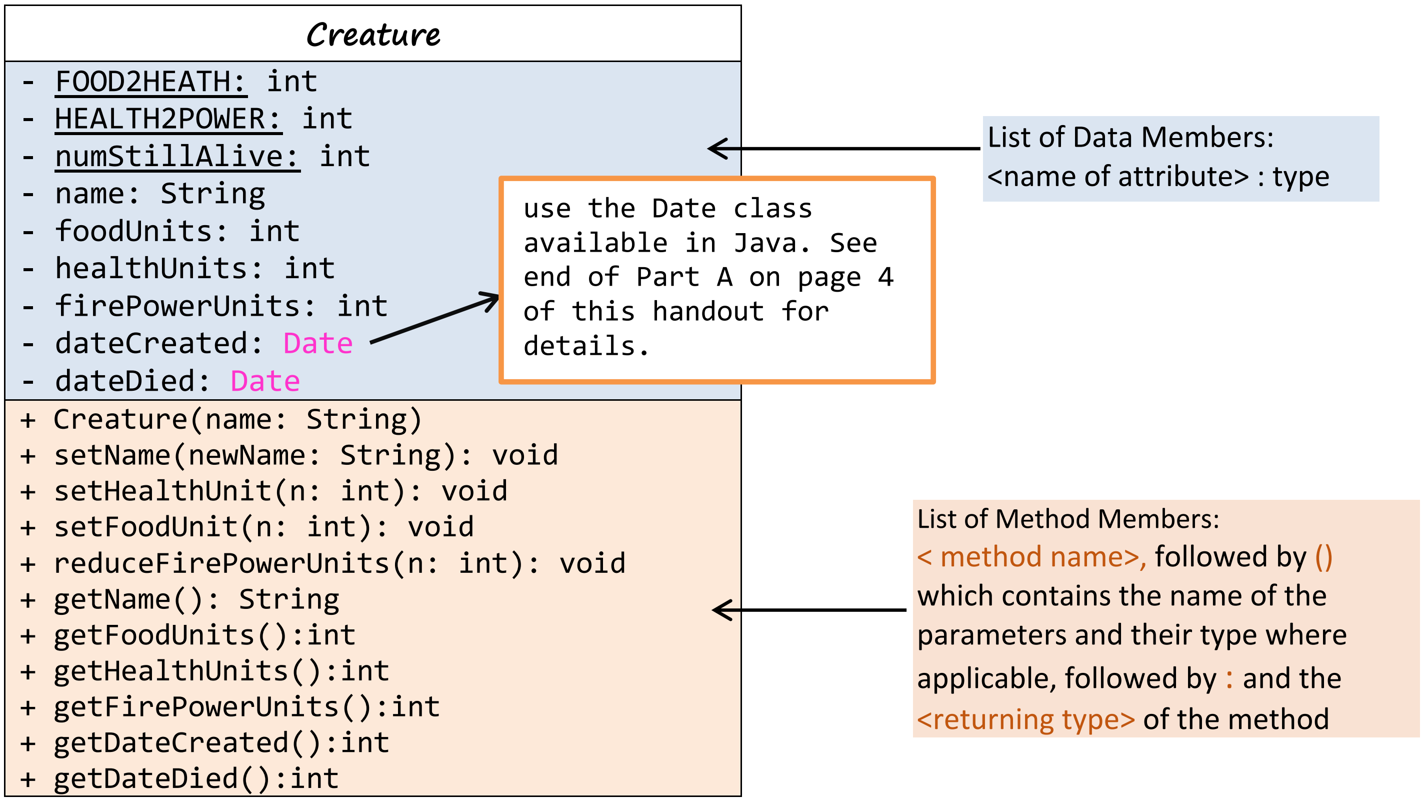
Due Date:	By 11:59pm Friday April 19, 2021
Evaluation:	8% of final mark (see marking rubric at the end of handout)
Late Submission:	none accepted
Purpose:	The purpose of this assignment is to help you learn Java classes and arrays of objects.
CEAB/CIPS Attributes:	Design/Problem analysis/Communication Skills

General Guidelines When Writing Programs:
Refer to Assignment #1 handout.

Part A – Preparing for the Battle Game Application

Write in JAVA the class ***Creature*** and test it before you tackle Part B. You can test the methods in the class by writing a driver and calling each method and making sure they behave as you intended them to.

The class ***Creature*** stores information about a player of the ***Battle Game*** (details in Part B). Following is a graphical summary of the class using UML (Unified Modeling Language) to give an overview of the class. A detailed description of each CLASS member follows.



```
Creature
- FOOD2HEATH: int
- HEALTH2POWER: int
- numStillAlive: int
- name: String
- foodUnits: int
- healthUnits: int
- firePowerUnits: int
- dateCreated: Date
- dateDied: Date
+ Creature(name: String)
+ setName(newName: String): void
+ setHealthUnit(n: int): void
+ setFoodUnit(n: int): void
+ reduceFirePowerUnits(n: int): void
+ getName(): String
+ getFoodUnits():int
+ getHealthUnits():int
+ getFirePowerUnits():int
+ getDateCreated():int
+ getDateDied():int
```

use the Date class available in Java. See end of Part A on page 4 of this handout for details.

List of Data Members:
<name of attribute> : type

List of Method Members:
< method name>, followed by () which contains the name of the parameters and their type where applicable, followed by : and the <returning type> of the method

```
+ getNumStillAlive(): int
+ isAlive(): boolean
+ earnFood(): int
+ attacking(player: Creature): void
+ healthFoodUnitsZero(): boolean
- died(): void
+ toString(): String
+ showStatus(): String
+ normalizeUnits(): void
```

Legends:
- means private member (data or method)
+ means public member (data or method)
Underlined members (data or method) are static.

Description of the class ***Creature***: The class contains the following members

- Data Members: (Refer to the UML on the previous page for the type of the data members.)

    - Private static constants FOOD2HEALTH set to 6 and HEALTH2POWER set to 4.
    - Private instance variables which keep track of a creature's name, number of food, health and fire power units, the date the creature was created, and the date it died.
    - Private static data member (numStillAlive) which keeps track of the number of creatures which are alive initialized to 0.

- Method Members

    - Constructors: *The constructor must increment the* numStillAlive *attribute by 1.* The constructor takes a string as parameter and sets the name attribute to the parameter, the foodUnits attribute to a random number between 1 and 12 inclusive, of healthUnits attribute to a random number between 1 and 7 inclusive, the firePowerUnits attribute to a random number between 0 and 10, the dateCreated to the current date (java.util.Date) in the system and the dateDied to null. **Remember to call normalizeUnits() after initializing health/food/fire power units.**

    - A public mutator method to set the name of the creature to the value passed.
    - A public mutator method to set the number of health units.
    - A public mutator method to set the number of food units.
    - A public method called reduceFirePowerUnits() which will reduce the fire power units by the amount specified by the parameter.
    - Public accessor methods for each non-static instance variable.
    - A public static accessor method to get the static instance variable numStillAlive.
    - A public method isAlive() that returns true if the dateDied is null and false otherwise.
    - A public method earnFood() which adds a number (randomly generated between 0 to 15 inclusive) of units to foodUnits. It then calls on the private normalizeUnits() method (see description below).
    - A public method attacking(player): This creature is attacking the player (the passed parameter). The attacking creature gets 50% of the other creature's food units and, 50% of its health units (**i.e., divide by 2 and round up**). Of course, this means that the

creature being attacked loses these units. The cost of attacking is 2 power units (reduce by 2 the firePowerUnits). It then calls the private normalizeUnits() method (see description below) to normalize the food, health and fire power units of the attacking creature. This method also calls the being-attacked-creature's (player) healthFoodUnitsZero() to determine whether the creature being attacked is dead after the attack.

- Public method healthFoodUnitsZero() that returns true if the creature has no more health and food units left, otherwise false. If no more health and food unit left, and the dateDied is null, call died().

- Private method died(), which sets the dateDied field to the current date.

- A public method toString( ) which takes no arguments and returns a string with all of the non-static attributes of the calling object in a descriptive message.
- A public method showStatus() which just returns the number of the food, health and fire power units as a String in the format:

```
"<foodUnits> food units, <healthUnits> health units,
<firePowerUnits> fire power units"
```
where <foodUnits> is the value stored in the attribute foodUnits, <healthUnits> is the value stored in the attribute healthUnits and <firePowerUnits> is the value stored in the attribute firePowerUnits.

- A private void method normalizeUnits()
  The method normalizeUnits() will convert each 6 units of food to a health unit and each 4 health units to a fire power unit. Note that on a rare condition, both health unit and food unit may both become 0 after a regular normalization, however this should be avoided, i.e., the food units are converted to health units entirely, but health units should be at least 4, e.g., 6 food units, 3 health units, and 1 fire power unit, should become 0 food unit, 4 health unit, and 1 fire power unit.

For Example: if a creature has 5 food units, 2 health units and 3 fire power units and gets an additional 8 food units, then the creature has 13 health units, 2 health units and 3 fire power units. After the normalization process, the creature will have 1 food units (12 gets converted to 2 health unit so one food unit left), 4 health units (since adding 2 health units) which will then become 0 health units, (since 4 health units get converted to 1 fire power unit) and 4 fire power units (since adding a fire power unit).

Test the methods in this class by writing a testing driver (which you are not required to submit) before moving on to Part B.

Part B – Now Ready to Play the Battle Game Application

Write a driver in JAVA to play the BATTLE GAME such that your program:

1. Displays a welcome message of your choice.
2. Creates the characters:
   I. Prompt the user for the number of creatures which will play the game. Make sure it is between 2 and 8 inclusive). See Figure 1.
   II. Create an array containing the requested number of objects of type ***Creature.*** Display the content after it is created. **Note**: the name of a creature can be more than one word. See Figure 2.

```
[----------------------------------------------]
[ Welcome to the Battle Game ]
[----------------------------------------------]
How many creatures would you like to have (minimum 2, maximum 8)? 1
*** Illegal number of creatures requested ***
How many creatures would you like to have (minimum 2, maximum 8)? 2
```

Figure 1- Prompt for number of players/creatures

```
What is the name of creature 1? Kokomo

Food units          Health units            Fire power units             Name
----------          ------------            ----------------             ----
4                   2                       3                            Kokomo
Date Create: Mon Mar 15 18:05:20 EDT 2021
Date died: is still alive

What is the name of creature 2? Sparki

Food units          Health units            Fire power units             Name
----------          ------------            ----------------             ----
2                   0                       3                            Sparki
Date Create: Mon Mar 15 18:06:40 EDT 2021
Date died: is still alive
```

Figure 2- Create the creatures

3. Play the game:

Randomly decide which creature starts by generating a random number between 1 and the number of creatures in the game. Each creature plays in sequence. So if there are 4 creatures and the third creature goes 1st, then the sequence of play is 3, 4, 1, 2, 3, 4, ...

During its turn a creature has 6 possible choices. Some choices end the current players turn, while others allow them to choose additional actions. When prompting the creature for a move, display a message containing the creature's index number (in figure 3 is 1), its name (in figure 3 is Kokomo) and in the following choices: If a creature enters an invalid menu choice keep prompting it until it enters a valid choice (see Figure 3).

When Option 1 is selected:

Display the number of creature's still alive. This will require a call to the static accessor method numStillsAlive(). It is still the current creature's turn. See Figure 4.

```
Creature #1: Kokomo, what do you want to do?
        1. How many creatures are alive?
        2. See my status
        3. See status of all players
        4. Change my name
        5. Work for some food
        6. Attack another creature (Warning! may turn against you)
Your Choice please > 8

Creature #1: Kokomo, what do you want to do?
        1. How many creatures are alive?
        2. See my status
        3. See status of all players
        4. Change my name
        5. Work for some food
        6. Attack another creature (Warning! may turn against you)
Your Choice please >
```

Figure 3- Screen capture of illegal menu choice

```
Creature #1: Kokomo, what do you want to do?

        1.How many creatures are alive?
        2. See my status
        3. See status of all players
        4. Change my name
        5. Work for some food
        6. Attack another creature (Warning! may turn against you)

Your Choice please > 1
     Number of creatures alive 2
```

Figure 4 - Screen capture of Option 1

When Option 2 is selected:

Displays the current player's (creature's) information. It is still the current creature's turn. See Figure 5.

```
Creature #1: Kokomo, what do you want to do?

        1.How many creatures are alive?
        2. See my status
        3. See status of all players
        4. Change my name
        5. Work for some food
        6. Attack another creature (Warning! may turn against you)
Your Choice please > 2
Food units          Health units           Fire power units            Name
----------          ------------           ----------------            ----
4                   2                      3                           Kokomo
Date Create: Mon Mar 15 18:05:20 EDT 2021
Date died: is still alive
```

Figure 5- Screen capture of Option 2

When Option 3 is selected:
Displays to the screen the information for all creatures, (all attributes except for the static attribute). It is still the current creature's turn (see Figure 6).

When Option 4 is selected:

Prompt the user for a new name and change the current creature's name to the new name. A name should be able to contain spaces. It is still the current creature's turn (see Figure 7).

```
Creature #1: Kokomo, what do you want to do?

       1.How many creatures are alive?
       2. See my status
       3. See status of all players
       4. Change my name
       5. Work for some food
       6. Attack another creature (Warning! may turn against you)
Your Choice please > 3
Food units          Health units            Fire power units             Name
----------          ------------            ----------------             ----
4                   2                       3                            Kokomo
Date Create: Mon Mar 15 18:05:20 EDT 2021
Date died: is still alive
Food units          Health units            Fire power units             Name
----------          ------------            ----------------             ----
2                   0                       3                            Sparki
Date Create: Mon Mar 15 18:06:40 EDT 2021
Date died: is still alive
```

Figure 6 - Screen capture of Option 3

```
Creature #1: Kokomo, what do you want to do?

       1.How many creatures are alive?
       2. See my status
       3. See status of all players
       4. Change my name
       5. Work for some food
       6. Attack another creature (Warning! may turn against you)
Your Choice please > 4
       Your name is currently "Kokomo"
       What is the new name: Toutou
```

Figure 7- Screen capture of Option 4

When Option 5 is selected: Work to earn extra food units (call earnFood() method). Instead of attacking another creature, the creature decides to refuel its food units to be a stronger creature. Current creature's turn is over and control passes to the next creature which is still alive. (Suggestion: display the units of the current creature, the number of units earned and the new disribution of units to make sure your program is working correctly). See Figure 8. (Notice that Creature #1's name is now Toutou)

```
Creature #1: Toutou, what do you want to do?
        1. How many creatures are alive?
        2. See my status
        3. See status of all players
        4. Change my name
        5. Work for some food
        6. Attack another creature (Warning! may turn against you)

Your Choice please > 5

Your status before working for food: 4 food units, 2 health units, 3 fire power
units ... You earned 4 food units.

Your status after working for food: 2 food units, 3 health units, 3 fire power
units
```

Figure 8- Screen capture of Option 5

When **Option 6** is selected:
The creature is prompted for the index number of the creature it wishes to attack. Of course you don't want to attack yourself and you cannot attack a creature that is no longer alive. If the user enters the index of a dead creature, of itself or an index that is out of bounds keep prompting the user until a valid index is entered (see Figure 9).

The creature at play has 1 in 3 chances of being attacked instead of attacking (generate a random number from 0 to 2, if it is 0 the creature at play is being attacked by the chosen creature otherwise the creature at play is attacking the chosen creature). The method `attacking()` is then called. (Suggestion: display the units of the current creature, the number of units earned and the new distribution of units to make sure your program is working correctly). Current creature's turn is over and control passes to the next creature which is still alive.

Remember to attack a creature must have at least 2 `firePower` units. If you do not have 2 `firePower` units and initialize the attack, you will end up being attacked by the creature you chose to attack (see Figure 10). If the creature you choose also does not have 2 `firePower` units neither, display "`Lucky you, the odds were that the other player attacks you, but <the name of the creature you choose to attack> doesn't have enough fire power to attack you! So is status quo!!`".

Finally, when there is only one creature alive, declare "GAME OVER" and display the final status of all creatures (see Figure 11).

```
Creature #2: Sparki, what do you want to do?

        1. How many creatures are alive?
        2. See my status
        3. See status of all players
        4. Change my name
        5. Work for some food
        6. Attack another creature (Warning! may turn against you)
   Your Choice please > 6

Who do you want to attack? (enter a number from 1 to 2 other than yourself(2): 4
That creature does not exist. Try again ...

Who do you want to attack? (enter a number from 1 to 2 other than yourself(2): 2
Can't attack yourself silly! Try again ...

Who do you want to attack? (enter a number from 1 to 2 other than yourself(2): 1

..... You are attacking Toutou!
Your status before attacking: 2 food units, 0 health units, 3 fire power units
Your status after attacking: 3 food units, 2 health units, 1 fire power units
```

Figure 9- Screen capture of Option 6 (example 1) – Case of a successful attack

```
Creature #2: Sparki, what do you want to do?

        1. How many creatures are alive?
        2. See my status
        3. See status of all players
        4. Change my name
        5. Work for some food
        6. Attack another creature (Warning! may turn against you)
   Your Choice please > 6

Who do you want to attack? (enter a number from 1 to 2 other than yourself(2): 1
That was not a good idea ... you only had 1 Fire Power units!!!
....... Oh No!!! You are being attacked by Toutou!
Your status before being attacked: 3 food units, 2 health units, 1 fire power units
Your status after being attacked: 1 food units, 1 health units, 1 fire power units
```

Figure 10- screen capture of Option 6 (example 2) - Did not have enough power unit and end up being attacked. Note that the data members of Sparki may not be consistent with previous figures, this is for illustration only.

```
Creature #1: Toutou, what do you want to do?
      1. How many creatures are alive?
      2. See my status
      3. See status of all players
      4. Change my name
      5. Work for some food
      6. Attack another creature (Warning! may turn against you)

Your Choice please > 6

Who do you want to attack? (enter a number from 1 to 2 other than yourself(1): 2

..... You are attacking Sparki!
Your status before attacking: 2 food units, 0 health units, 3 fire power units
Your status after attacking: 3 food units, 1 health units, 1 fire power units

---->Sparki is dead

GAME OVER!!!!!

Food units          Health units            Fire power units             Name
----------          ------------            ----------------             ----
3                   1                       31                           Toutou
Date Create: Mon Mar 15 18:05:20 EDT 2021
Date died: is still alive
Food units          Health units            Fire power units             Name
----------          ------------            ----------------             ----
0                   0                       1                            Sparki
Date Create: Mon Mar 15 18:06:40 EDT 2021
Date died: Mon Mar 15 18:08:30 EDT 2021
```

Figure 11-Screen capture of Option 6 (example 3), after a successful attack, Creature #2 Sparki is dead. Since there is only one alive creature, the game is over. Display the final status of all creatures. Sparki's data is based on Figure 10.

Submitting Assignment 4

Please check your course webpage on how to submit the assignment.

Evaluation Criteria for Assignment 4 (20 points)

Source Code	
Comments for (3 pts.)	
Description of the program (authors, date, purpose)	0.5 pts.
Description of variables and constants	1 pt.
Description of the algorithm	1.5 pts.
Programming Style for all 2 questions (3 pts.)	
Use of significant names for identifiers	1 pt.
Indentation and readability	1 pt.
Welcome Banner or message/Closing message	1 pt.
Part A (6 pts.)	
Correct implementation of data members	1.5 pt.
Correct implementation of method members	3.5 pts.
Display the correct result	1 pt.
Part B (8 pts.)	
Prompt user’s inputs	1 pt.
Correctly initializing the arrays of Creatures	1 pt
Correct implementation of the gaming logics	4 pts.
Display result	2 pts.
TOTAL	**20 pts.**