

Project 1

● Graded

Student

Devin Chen

Total Points

100 / 100 pts

Autograder Score

80.0 / 80.0

Passed Tests

Test compiles (5/5)

Tests Creature class default constructor (10/10)

Tests Creature class parameterized constructor with all arguments given (10/10)

Tests Character class parameterized constructor with default arguments (10/10)

Tests Creature class mutator functions (25/25)

Tests Creature class display function (5/5)

Checks that test.cpp tests Creature functions (10/10)

Test for read-only functions and parameters (5/5)

Question 2

Style & Documentation

20 / 20 pts

✓ + 5 pts Style

✓ + 5 pts Indicates name and description in comment preamble at top of file

✓ + 10 pts Has function preambles with @pre, @post, @param, @return where appropriate

+ 20 pts No-Compile Adjustment

+ 0 pts Insufficient submission

Autograder Results

Test compiles (5/5)


Your program compiles!

Tests Creature class default constructor (10/10)

Your program passed.

Tests Creature class parameterized constructor with all arguments given (10/10)
Your program passed.
Tests Character class parameterized constructor with default arguments (10/10)
Your program passed.
Tests Creature class mutator functions (25/25)
Your program passed.
Tests Creature class display function (5/5)
Your program passed.
Checks that test.cpp tests Creature functions (10/10)
test.cpp compiles! Your program is testing the Creature class appropriately.
Test for read-only functions and parameters (5/5)
Your program has functions and parameters as read-only (const) where appropriate.

Submitted Files

▼ .gitignore	 Download
1	.DS_Store
2	.vscode
3	*.log
4	
5	

```
1  /**
2   * @file Creature.hpp
3   * @author Devin Chen
4   * @brief Creature Class
5   * @date 1/20/2024
6   */
7
8
9  #include "Creature.hpp"
10
11 /**
12  * Default constructor.
13  * Default-initializes all private members.
14  * Default creature name: "NAMELESS".
15  * Booleans are default-initialized to False.
16  * Default enum value: UNKNOWN
17  * Default Hitpoints and Level: 1.
18  */
19 Creature::Creature():name_{"NAMELESS"},category_{UNKNOWN},
hitpoints_{1},level_{1},is_tame_{false}{};
20
21 /**
22  * Parameterized constructor.
23  * @param    : A reference to the name of the creature (a string). Set the creature's name to
NAMELESS if the provided string contains non-alphabetic characters.
24  * @param    : The category of the creature (a Category enum) with default value UNKNOWN
25  * @param    : The creature's hitpoints (an integer) , with default value 1 if not provided, or if the
value provided is 0 or negative
26  * @param    : The creature's level (an integer), with default value 1 if not provided, or if the value
provided is 0 or negative
27  * @param    : A flag indicating whether the creature is tame, with default value False
28  * @post     : The private members are set to the values of the corresponding parameters. The
name is converted to UPPERCASE if it consists of alphabetical characters only, otherwise it is set to
NAMELESS.
29  */
30 Creature::Creature(const std::string &new_name, Category new_category, int new_hitpoint, int
new_level, bool new_tame){
31     if(!SetName(new_name)) {
32         name_ = "NAMELESS";
33     }
34     setCategory(new_category);
35     if(!setLevel(new_level)){
36         level_ = 1;
37     }
38     if(!setHitpoints(new_hitpoint)){
39         hitpoints_ = 1;
40     }
41     setTame(new_tame);
42 }
```

```

43
44
45 /**
46  * @param : the name of the Creature, a reference to string
47  * @post : sets the Creature's name to the value of the parameter in UPPERCASE.
48  * (convert any lowercase character to uppercase)
49  * Only alphabetical characters are allowed.
50  * : If the input contains non-alphabetic characters, do nothing.
51  * @return : true if the name was set, false otherwise
52 */
53 bool Creature::setName(const std::string &new_name){
54     for(int i = 0; i < new_name.length(); i++){
55         if (!isalpha(new_name[i])){
56             return false;
57         }
58     };
59     std::string tempname;
60     for (int i = 0; i < new_name.length(); i++){
61         tempname += toupper(new_name[i]);
62     }
63     name_ = tempname;
64     return true;
65 };
66
67 /**
68  * @return : the name of the Creature
69 */
70 std::string Creature::getName() const{
71     return name_;
72 }
73
74 /**
75  * @param : a reference to Category, the category of the Creature (an enum)
76  * @post : sets the Creature's category to the value of the parameter
77  * : If the given category was invalid, set category_ to UNKNOWN.
78 */
79 void Creature::setCategory(const Category &new_category){
80     if(new_category >= UNKNOWN && new_category <= ALIEN) {
81         category_ = new_category;
82     }
83     else{
84         category_ = UNKNOWN;
85     }
86 }
87 /**
88  * @return : the category of the Creature (in string form)
89 */
90 std::string Creature::getCategory() const{
91     switch(category_){
92         case UNKNOWN:
93             return "UNKNOWN";
94         case UNDEAD:

```

```
95         return "UNDEAD";
96     case MYSTICAL:
97         return "MYSTICAL";
98     case ALIEN:
99         return "ALIEN";
100 }
101 }
102
103 /**
104  * @param : a reference to integer that represents the creature's hitpoints
105  * @pre : hitpoints >= 0 : Characters cannot have negative hitpoints
106  * (do nothing for invalid input)
107  * @post : sets the hitpoints private member to the value of the parameter
108  * @return : true if the hitpoints were set, false otherwise
109 */
110 bool Creature::setHitpoints(const int &new_hitpoint){
111     if (new_hitpoint <= 0){
112         return false;
113     }
114     hitpoints_ = new_hitpoint;
115     return true;
116 }
117
118 /**
119  * @return : the value stored in hitpoints_
120 */
121 int Creature::getHitpoints() const{
122     return hitpoints_;
123 }
124
125 /**
126  * @param : a reference to integer level
127  * @pre : level >= 0 : Characters cannot have a negative level
128  * @post : sets the level private member to the value of the parameter
129  * (do nothing for invalid input)
130  * @return : true if the level was set, false otherwise
131 */
132 bool Creature::setLevel(const int &new_level){
133     if (new_level <= 0){
134         return false;
135     }
136     level_ = new_level;
137     return true;
138 }
139
140 /**
141  * @return : the value stored in level_
142 */
143 int Creature::getLevel() const{
144     return level_;
145 }
146
```

```
147 /**
148  * @param : a reference to boolean value
149  * @post : sets the tame flag to the value of the parameter
150  */
151 void Creature::setTame(const bool &new_tame){
152     is_tame_ = new_tame;
153 }
154
155 /**
156  * @return true if the creature is tame, false otherwise
157  * Note: this is an accessor function and must follow the same convention as all accessor functions
158  * even if it is not called getTame
159  */
160 bool Creature::isTame() const{
161     return is_tame_;
162 }
163
164 /**
165  * @post : displays Creature data in the form:
166  * "[NAME]\n
167  * Category: [CATEGORY]\n
168  * Level: [LEVEL]\n
169  * Hitpoints: [Hitpoints]\n
170  * Tame: [TRUE/FALSE]"
171  */
172 void Creature::display() const{
173     std::cout << name_ << "\n";
174     std::cout << "Category: " << getCategory() << "\n";
175     std::cout << "Level: " << level_ << "\n";
176     std::cout << "Hitpoints: " << hitpoints_ << "\n";
177     if(is_tame_){
178         std::cout << "Tame: TRUE";
179     }
180     else {
181         std::cout << "Tame: FALSE";
182     }
183 }
```

```
1  /**
2   * @file Creature.hpp
3   * @author Devin Chen
4   * @brief Creature Class
5   * @date 1/20/2024
6   */
7
8  #pragma once
9  #include <iostream>
10 #include <string>
11 #include <cctype>
12
13 class Creature{
14 public:
15     enum Category {UNKNOWN, UNDEAD, MYSTICAL, ALIEN};
16
17 private:
18     std::string name_;
19     Category category_;
20     int hitpoints_;
21     int level_;
22     bool is_tame_;
23
24 public:
25
26  /**
27   * Default constructor.
28   * Default-initializes all private members.
29   * Default creature name: "NAMELESS".
30   * Booleans are default-initialized to False.
31   * Default enum value: UNKNOWN
32   * Default Hitpoints and Level: 1.
33   */
34  Creature();
35
36  /**
37   * Parameterized constructor.
38   * @param    : A reference to the name of the creature (a string). Set the creature's name to
39   NAMELESS if the provided string contains non-alphabetic characters.
40   * @param    : The category of the creature (a Category enum) with default value UNKNOWN
41   * @param    : The creature's hitpoints (an integer) , with default value 1 if not provided, or if the
42   value provided is 0 or negative
43   * @param    : The creature's level (an integer), with default value 1 if not provided, or if the value
44   provided is 0 or negative
45   * @param    : A flag indicating whether the creature is tame, with default value False
46   * @post     : The private members are set to the values of the corresponding parameters. The
47   name is converted to UPPERCASE if it consists of alphabetical characters only, otherwise it is set to
48   NAMELESS.
49   */
50 }
```

```

45 Creature(const std::string &new_name, Category new_category = UNKNOWN, int new_hitpoint = 1,
46 int new_level = 1 , bool new_tame = false);
47 /**
48  * @param : the name of the Creature, a reference to string
49  * @post : sets the Creature's name to the value of the parameter in UPPERCASE.
50  * (convert any lowercase character to uppercase)
51  * Only alphabetical characters are allowed.
52  * : If the input contains non-alphabetic characters, do nothing.
53  * @return : true if the name was set, false otherwise
54  */
55 bool setName(const std::string &new_name);
56
57 /**
58  * @return : the name of the Creature
59  */
60 std::string getName() const;
61
62 /**
63  * @param : a reference to Category, the category of the Creature (an enum)
64  * @post : sets the Creature's category to the value of the parameter
65  * : If the given category was invalid, set category_ to UNKNOWN.
66  */
67 void setCategory(const Category &new_category);
68
69 /**
70  * @return : the category of the Creature (in string form)
71  */
72 std::string getCategory() const;
73
74 /**
75  * @param : a reference to integer that represents the creature's hitpoints
76  * @pre : hitpoints >= 0 : Characters cannot have negative hitpoints
77  * (do nothing for invalid input)
78  * @post : sets the hitpoints private member to the value of the parameter
79  * @return : true if the hitpoints were set, false otherwise
80  */
81 bool setHitpoints(const int &new_hitpoint);
82
83
84 /**
85  * @return : the value stored in hitpoints_
86  */
87 int getHitpoints() const ;
88
89 /**
90  * @param : a reference to integer level
91  * @pre : level >= 0 : Characters cannot have a negative level
92  * @post : sets the level private member to the value of the parameter
93  * (do nothing for invalid input)
94  * @return : true if the level was set, false otherwise
95  */

```



```
96  bool setLevel(const int &new_level);
97
98  /**
99   * @return : the value stored in level_
100 */
101  int getLevel() const;
102
103  /**
104   * @param : a reference to boolean value
105   * @post  : sets the tame flag to the value of the parameter
106 */
107  void setTame(const bool &new_tame);
108
109  /**
110   * @return true if the creature is tame, false otherwise
111   * Note: this is an accessor function and must follow the same convention as all accessor functions
112   even if it is not called getTame
113 */
114  bool isTame() const;
115
116  /**
117   * @post   : displays Creature data in the form:
118   * "[NAME]\n
119   * Category: [CATEGORY]\n
120   * Level: [LEVEL]\n
121   * Hitpoints: [Hitpoints]\n
122   * Tame: [TRUE/FALSE]"
123 */
124  void display() const;
125  };
```

▼ Makefile

 Download

```
1 CXX = g++
2 CXXFLAGS = -std=c++17 -g -Wall -O2
3
4 PROG ?= main
5 OBJS = Creature.o test.o
6
7 all: $(PROG)
8
9 .cpp.o:
10     $(CXX) $(CXXFLAGS) -c -o $@ $<
11
12 $(PROG): $(OBJS)
13     $(CXX) $(CXXFLAGS) -o $@ $(OBJS)
14
15 clean:
16     rm -rf $(EXEC) *.o *.out main
17
18 rebuild: clean all
19
```

▼ README.md

 Download

```
1 \[!\[Review Assignment Due Date\]\(https://classroom.github.com/assets/deadline-readme-button-24ddc0f5d75046c5622901739e7c5dd533143b0c8e959d652212380cedb1ea36.svg\)\]
2 \[https://classroom.github.com/a/kuHzDhWw\]
3
4 # Project1
5
6 The project specification can be found on Blackboard
```

▼ test.cpp

 Download

```
1 #include "Creature.hpp"
2
3 int main(){
4
5     Creature dragon;
6     dragon.setHitpoints(10);
7     dragon.setLevel(5);
8     dragon.setTame(true);
9     dragon.display();
10
11     Creature worm("wormy",Creature::Category::MYSTICAL, 3,2);
12     worm.setTame(true);
13     worm.display();
14 }
```