Project 2 Design + Reflection
Tay, Serena
Student #: 933588509

Document (Design + Reflection)

You need to create your program design **BEFORE** you start coding. Your design needs to be included in your document.

Simply stated, **program design** is identifying the problem to be solved, modularizing and describing the structure of program, and making a test table to that fulfills the requirements of the program.

For test table, please include test plan, expected output, and actual output. The test plan should be robust. The expected output is based on the requirement. If your actual output deviates from your expected output, that usually means the program behavior is deviating from the requirements. Following is a Examples Test Plan.pdf

In your **reflection**, you should explain what design changes you made, how you solved some problems you encountered, and what you learned for this project. The TA will grade on how well your implementation matches your design.

Implemented the extra credit option #2:
Status messages: (5pts)

Design

Create three classes with the following functions

- Game class – As the 'master class' that dictates the flow of the program
- Menu class – To display initial menu and a prompt function that prompt users for inputs for the program
- Zoo class – to consolidate all the animal classes information and include functions to update age, create dynamic arrays and delete arrays whenever needed
- Animal class – parent class for all animals in the zoo
- Penguin class
- Tiger class
- Turtle class

Animal Class
Contain the following member variables to store user inputs and board data:

```
int age,
    cost,
    numberOfBabies,
    payoff,
    baseFoodCost;
```

Project 2 Design + Reflection
Tay, Serena
Student #: 933588509

Contain the following class functions:
Animal(); //default constructor to initialize all member variables

Animal(int age, int cost, int numberOfBabies, int payoff, int baseFoodCost) //to be used by inherited classes
void growOld(); //to add 1 day to age
int getAge(); //to get age to test if animal is old enough to make babies
int getCost(); //get cost of animal

Data validation/integrity consideration for Animal class:

- Nothing as this is the parent class that does not get any user inputs

Menu Class
Contain the following member variables:

- countPlay – This is used to determine if the "new run" menu should be used or "play again" menu should be used per requirements
- menuChoice – used to store choice entered by user

Contain the following member functions:

- Menu () – default constructor to initialize member variables used
- displayMenu() – to display menu (contained 2 menus, one for the new game, one for the 'play again' scenario once the first Game run occurred
- prompt(string, int, int) – a generic function that can be used to prompt user for data inputs. Contain 3 parameters, one string to tell user which member variable the program is asking for, one integer for the minimum allowable input and one for the maximum allowable input
- int promptChoice(string, string, string, string, string) – prompt when there are more than 1 choice available
- getMenuChoice() – function to be used in Game class to obtain user's input

Data validation/integrity consideration for Menu class:

- make sure user gets an error message when entering an invalid choice
- create a max and min for a range of allowable input (create an error message when user entered an invalid number outside the range)
- make sure non-ints are guarded from creating errors in the program

Game class
Contain the following member variables:

```
int day,
    penguinBuy,
```

Project 2 Design + Reflection
Tay, Serena
Student #: 933588509

```
    tigerBuy,
    turtleBuy,
    choice;
```

the buy variables are used to store user inputs for purchases made during the day.

Contain the following member functions:

- Game() – default constructor used to initialize all the member variables listed above
- runGame() – used to run the Game and calling the menu and Animal class in sequential order (display menu, ask user for input, run program, ask user what they wAnimal and repeat until user want to quit)
- void randomEvents() – to run the random selection of daily random events
- void firstBuy(); - to call functions from zoo class to create objects, update bank balance and animal count for first purchase
- void dailyBuy(); - to call functions from zoo class to create objects, update bank balance and animal count for all daily purchase

The following member functions initiate the four random daily events based on requirements set:
void randomEvent1();
void randomEvent2();
void randomEvent3();
void randomEvent4();

Data validation/integrity consideration for Game class:

- Use menu functions to ensure user inputs are valid for daily purchase, first purchase and choice to move to next day or quit

Penguin/Tiger/Turtle (please note that all three inherited classes have the same member functions declared but different names based on the animal type):
Penguin(); //initial constructor to initialize inherited member variables to specific values for each animals
Penguin(int); //for babies and daily purchases (adults that need to start at age 3)

Zoo class
Member variables include the following:

```
// three dynamic arrays to store objects of each animal type
Penguin *penguinArray;
Tiger *tigerArray;
Turtle *turtleArray;
int bankBalance,
    //counter to keep track of each animal type counts
```

Project 2 Design + Reflection
Tay, Serena
Student #: 933588509

```
      penguinCount,
      tigerCount,
      turtleCount,
      //used to check capacity to determine if resizing is needed
      penguinCapacity,
      tigerCapacity,
      turtleCapacity;

public:
  //data of each animals are created so game class can access them but not change them
  static const int costPenguin = 1000;
  static const int costTiger = 10000;
  static const int costTurtle = 100;
  static const int baseFoodCost = 10;
  static const int foodCostPenguin = baseFoodCost;
  static const int foodCostTiger = 5 * baseFoodCost;
  static const int foodCostTurtle = (baseFoodCost / 2);
```

public member vars are “proctected” by making them static constants to ensure the game class have access to them for calculating food cost and purchase costs.

Member functions to be used during the daily game flow update routines:

Random event specific functions
 int bonusPayoff();
void removeSickAnimal(int);
 bool checkAdult(int);
 int getAnimalAge(int);

Animal objects and resizing of arrays
 void updateAnimalCount(int, int);
 void createAnimal(int, int, int);
 int getAnimalCount(int);
 void resizeArray(int, int);
 void increaseCapacityCounter(string);
 void deletePointers();

General game flow functions
 void updateAge();
 void updateBankBalance(int);
 void payFoodCost();
 void printDailyReport(int);
 bool checkBankBalance();

Project 2 Design + Reflection
Tay, Serena
Student #: 933588509

Project 2 Design + Reflection
Tay, Serena
Student #: 933588509

Testing plan and results

Menu class

- displayMenu() – to display menu (contained 2 menus, one for the new game, one for the 'play again' scenario once the first Game run occurred

Test Case	Input Values	Expected Outcomes	Observed Outcome
Display 1st menu when program initialize	Menu counter	Display first menu	Success – Achieved expected outcomes
Display "play again menu" after first run	Menu counter	Display play again menu	Success – Achieved expected outcomes

- prompt(string, int, int) – a generic function that can be used to prompt user for data inputs. Contain 3 parameters, one string to tell user which member variable the program is asking for, one integer for the minimum allowable input and one for the maximum allowable input

Test Case	Input Values	Expected Outcomes	Observed Outcome
Display correct output for all prompts used in Game class	Game class calls	Display appropriate prompts for: 1. # of animals for first day purchase 2. Purchase made for all days	Success – Achieved expected outcomes for all inputs prompted for
Display error messages if input does not fall between the min and max set	Game class calls	Display appropriate error message when user input is outside of valid range set and reprompt user until valid input is provided: 1. # of animals for first day purchase 2. Purchase made for all days	Success – Achieved expected error message outcomes for all invalid inputs and reprompt user for valid input

Game class

- Game() – default constructor used to initialize all the member variables listed above

Test Case	Input Values	Expected Outcomes	Observed Outcome

Project 2 Design + Reflection
Tay, Serena
Student #: 933588509

Ensure constructor initialize to default values	N/A – set in program	Member variables initialize to the values stated	Member variables initialized to the values stated

- runGame() – used to run the Game and calling the menu and Animal class in sequential order (display menu, ask user for input, run program, ask user what they want and repeat until user want to quit)

Test Case	Input Values	Expected Outcomes	Observed Outcome
Ensure function is run the function calls in sequence and keep looping until user decides to quit	N/A – set in program	1. display menu 2. ask user for input (purchases) 3. run program 4. ask user if the 5. Repeat loop until user want to quit	Success – Achieved expected error message outcomes for all invalid inputs and re-prompt user for valid input

Test Case	Input Values	Expected Outcomes	Observed Outcome
Ensure functions are returning the right values when called in the Animal class	User input	Call returns the user input stored in the member variables	Success - Call returned the user input stored in the member variables

Zoo class

Test Case	Input Values	Expected Outcomes	Observed Outcome
Update animal count	Situations involved: 1. User inputs purchase made 2. Baby boom	- Add 1 or 2 based on whether user input 1 or 2 - Add count based on animal type - Make sure counts add to the correct animal type	Success – expected outcome achieved
updateBankBalance function is calculating properly	- Animal purchases - Bonus payoff - Food cost	- Deduct bank balance by the cost set in the animal classes	Success – changes to bank balance by function implemented correctly

Project 2 Design + Reflection
Tay, Serena
Student #: 933588509

		- Bonus added to balance correctly (and within the 250 to 500 per tiger limit) - Cost deducted correctly	
Ensure only animals with adults are allowed to make babies when selected	- Animal age for all 3 animal types	- Only one adult makes baby when selected	Success – expected outcome achieved
Resize array	- Current capacity	- Based on situation (purchase or baby birth), check array to ensure there is enough capacity and if not, double capacity	Success – expected outcome achieved
Memory leak	N/A	Run program with Valgrind to ensure no memory leak	Success – expected outcome achieved

Project 2 Design + Reflection
Tay, Serena
Student #: 933588509

Reflection

Inheritance

- I started the program without using much of virtuals and redeclaring new member variances in the inherited classes
- Since then I reformatted the program to leverage the capabilities of inheritance (reusing parent class member variables) and functions to initialize to the child classes values as well as obtaining the age in the zoo class

Design

- Initially I wanted to create a function to handle all four random events but figured from a design perspective, dividing all four random events into separate functions makes it easier to configure the function to select the random functions.
- To take into consideration some of the nuances such as animals that doesn't have adults or if there are no tigers for the random events, I have designed the random events to handle all the possible situation specifically (i.e. if there is only one animal available for baby boom to 3 animals available, if there are no tigers) the random functions would rerun to only include other viable events. This however was too extensive for the program and could be simplified in the future with a simple reroute to "nothing happened" or rerunning the random function until one of the event is acceptable is selected
- To ensure the program is running correctly, I initially had difficulty detecting if the program was calculating the count, capacity and bank balance correctly which are important to resize arrays and detecting if there is a negative bank balance. I created a print function in zoo class which outputs a report on a daily basis which tells the user what the status is for each day

Testing Phase

- I had difficulty keeping track of bank balances and capacity when testing the program and eventually created a testing table in excel to log the results and inputs to ensure the program is giving the correct expected output for bank balance, number of animals and the capacity available (see below)

Project 2 Design + Reflection
Tay, Serena
Student #: 933588509

4															
5				100,000			Total	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 8
6															
7	Penguin	Buy	4	(4,000)	(1,000)		27	2	2	2	2	2	3	3	3
8		Food		(270)	(10)										
9															
10															
11	Tiger	Buy	9	(90,000)	(10,000)		64	3	4	4	4	5	5	5	5
12		Food		(3,200)	(50)										
13		Bonus #1		1,875											
14				2,072											
15															
16	Turtle	Buy	4	(400)	(100)		69	2	2	2	3	3	3	4	4
17		Food		(345)	(5)										
18															
19			Per test	5,732											
20			Per program	5,732											
21			Diff	-											
22															
23															
24															
25		Tiger payoff (must be between 250 - 500 per tiger)													
26			Total	2,072											
27				230											