CSCE 145 Exam 2 Review

What to Expect

- Any material from Exam01
- Methods
- Classes
- Statics
- Inheritance

What will NOT be on the Exam

- Multidimensional arrays
- Switch statements
- For-each loops
- Enumerations
- Interfaces
Parts 5 and 6 Methods, Classes / Objects, and Arrays of Objects

1. There’s a Light (Part 01)

Objective:

Create a class **Light** that has the following properties and methods.

Requirements:

- **Functionality. (20pts)**
  - No Syntax, Major Run-Time, or Major Logic Errors. (20pts*)
    - *Code that cannot be compiled due to syntax errors is nonfunctional code and will receive no points for this entire section.
    - *Code that cannot be executed or tested due to major run-time or logic errors is nonfunctional code and will receive no points for this entire section.
    - *In addition, this also applies to the provided code, which must not be modified.
  - Properties / Instance Variables and Constants (5pts)
    - **isOn:** an instance variable that is either true or false depending on if the light is on or off respectively. The default value must be false.
    - **bulbWattage:** an instance variable that is a non-negative integer corresponding to the number of watts in the bulb. The default value should be 50.
    - Each instance variable must have the private scope and class constants must have the public scope.
    - All must apply for full credit.
  - **Constructors**
    - Default: set the instance variables to a default value as described in the “Properties / Instance Variables and Constants” section.
    - Parameterized: One that takes in two parameters that will set the instance variables and check for valid values. The order of these parameters needs to follow the order found in part the “Properties / Instance Variables and Constants” section. (IE the first point’s variable must occur first, then second point’s variable must occur next, and so on).
    - All must apply for full credit.
  - **Accessors and Mutators for Each Instance Variable**
    - Accessors and Mutators must follow the same structure that is used in lecture including but not limited to the scope, return types, identifiers, and parameters.
    - Mutators must check for valid values as described in the “Properties / Instance Variables and Constants” section.
    - All must apply for full credit.
• Coding Style. (2pts)
  o Readable Code
    ▪ Meaningful identifiers for data and methods.
    ▪ Proper indentation that clearly identifies statements within the body of a class, a method, a branching statement, a loop statement, etc.
    ▪ All the above must apply for full credit.
• Comments. (3pts)
  o Your name in the file. (1pts)
  o At least 5 meaningful comments in addition to your name. These must describe the function of the code it is near. (2pts)

Solution Tests:
• Does the solution compile?
• Does the solution have your name in the comments and 5 meaningful comments that describe the code?
• Does the output from the Question01 tester file match the following?

Constructing a new instance of Light using the Default Constructor
Testing Accessors
Is On: false
Bulb Wattage: 50
Testing Mutators with Correct Values
Is On: true
Bulb Wattage: 90
Testing Mutators with Incorrect Values
Bulb Wattage: 50
Constructing a new instance of Light using the Param Constructor
Is On: true
Bulb Wattage: 100
2. There’s a Light (Part 02)

Objective:

Continue creating a class Light that has the following methods.

Requirements:

- Functionality. (20pts)
  - No Syntax, Major Run-Time, or Major Logic Errors. (20pts*)
    - *Code that cannot be compiled due to syntax errors is nonfunctional code and will receive no points for this entire section.
    - *Code that cannot be executed or tested due to major run-time or logic errors is nonfunctional code and will receive no points for this entire section.
    - *In addition, this also applies to the provided code, which must not be modified.
  - toString Method.
    - This method takes in no parameters and returns a String with the values of the instance variables. It should return a String formatted as:
      
      “Is On: <<value of isOn>> Bulb Wattage: <<value of bulbWattage>>”

    - All the above must apply for full credit.
  - equals Method.
    - This method takes in an instance of Light and returns true only if all properties are equal.
    - This method must check if the other instance exists first, and if it does not then it must return false.
    - All the above must apply for full credit.
  - copyLight Method.
    - This static method takes in an instance of Light and returns a new instance of Light with the same properties.
    - This method must be defined as static.
    - This method must return a new instance of Light with the same properties as the other light.
    - All the above must apply for full credit.
  - turnOnLight Method.
    - This method returns no values and takes in no parameters.
    - This method sets the instance variable isOn to true.
    - All the above must apply for full credit.
  - turnOffLight Method.
    - This method returns no values and takes in no parameters.
    - This method sets the instance variable isOn to false.
All the above must apply for full credit.

- Coding Style. (2pts)
  - Readable Code
    - Meaningful identifiers for data and methods.
    - Proper indentation that clearly identifies statements within the body of a class, a method, a branching statement, a loop statement, etc.
  - All the above must apply for full credit.

- Comments. (3pts)
  - Your name in the file. (1pts)
  - At least 5 meaningful comments in addition to your name. These must describe the function of the code it is near. (2pts)

Solution Tests:
- Does the solution compile?
- Does the solution have your name in the comments and 5 meaningful comments that describe the code?
- Does the output from the Question02 tester file match the following?

Constructing a 2 instances of Light using the Default Constructor
Testing the toString Method
Is On: false Bulb Wattage: 50
Testing the toString turnOnLight and turnOffLight method
Is On: true Bulb Wattage: 50
Is On: false Bulb Wattage: 50
Testing equals Method with 2 lights that have the same properties
Are they equal? true
Setting values of Light 2 to have different values.
Testing equals Method with 2 lights that have different properties
Are they equal? false
Setting values of Light 2 to null.
Testing equals Method with 2 lights where one is null
Are they equal? false
Testing equals copyLight method by copying the first light
Are they equal? true
3. Lowest Priced Cars in the Midlands!

Objective:

Using the provided code (Car.java and Question03.java), write a method that returns the instance of a Car with the smallest price from the array of cars.

Requirements:

- Functionality. (20pts)
  - No Syntax, Major Run-Time, or Major Logic Errors. (20pts*)
    - *Code that cannot be compiled due to syntax errors is nonfunctional code and will receive no points for this entire section.
    - *Code that cannot be executed or tested due to major run-time or logic errors is nonfunctional code and will receive no points for this entire section.
    - *In addition, this also applies to the provided code, which must not be modified.
  - Method Signature / Definition.
    - This method must be written inside the provided code, and must have the identifier ###
    - This method must return an instance of a Car and this method requires no additional information.
    - All the above must apply for full credit.
  - Finding and Returning the Lowest Priced Car.
    - The method must return the instance of a Car that has the lowest price found in the provided instance variable array.
    - You may assume that the instance variable array has been constructed prior to calling the method.
    - You may NOT assume that every instance of Car in the array is constructed, and the program must check this to avoid a major run-time error.
    - If the array is empty, thus there are no constructed instances of Car in the array, then the method must return null.
    - If every car has the exact same price, then return a Car with the price.
    - All the above must apply for full credit.

- Coding Style. (2pts)
  - Readable Code
    - Meaningful identifiers for data and methods.
    - Proper indentation that clearly identifies statements within the body of a class, a method, a branching statement, a loop statement, etc.
    - All the above must apply for full credit.

- Comments. (3pts)
  - Your name in the file. (1pts)
At least 5 meaningful comments in addition to your name. These must describe the function of the code it is near. (2pts)

**Solution Tests:**

- Does the solution compile?
- Does the solution have your name in the comments and 5 meaningful comments that describe the code?
- Does the output from the Question03 tester file match the following?

**Constructing and Populating Car Array**
Testing the getLowestPrice method
Make: make07 Model: model07 Price: 4000.0
Resetting some of the Car's Prices.
Testing the getLowestPrice method with different values
Make: make02 Model: model02 Price: 1000.0
Testing getLowestPrice method with an empty Array
Does this return null?
true
Part 07 Inheritance and Polymorphism.

Objective:

Using the provided code (Person.java and Question04.java), create a class Employee that inherits from type Person and has the following.

Requirements:

- Functionality. (20pts)
  - No Syntax, Major Run-Time, or Major Logic Errors. (20pts*)
    - *Code that cannot be compiled due to syntax errors is nonfunctional code and will receive no points for this entire section.
    - *Code that cannot be executed or tested due to major run-time or logic errors is nonfunctional code and will receive no points for this entire section.
    - *In addition, this also applies to the provided code, which must not be modified.
  - Properties / Instance Variables and Constants (5pts)
    - employeeNumber: a non-negative, whole number value. The default value should be 0.
    - Each instance variable must have the private scope and class constants must have the public scope.
    - All must apply for full credit.
  - Constructors
    - Default: set the instance variables to a default value as described in the “Properties / Instance Variables and Constants” section.
    - Parameterized: One that takes in two parameters that will set the instance variables and check for valid values. The order of these parameters needs to follow the order found in part the “Properties / Instance Variables and Constants” section. (IE the first point’s variable must occur first, then second point’s variable must occur next, and so on).
    - All must apply for full credit.
  - Accessors and Mutators for Each Instance Variable
    - Accessors and Mutators must follow the same structure that is used in lecture including but not limited to the scope, return types, identifiers, and parameters.
    - Mutators must check for valid values as described in the “Properties / Instance Variables and Constants” section
    - All must apply for full credit.
  - toString method.
    - This method takes in no parameters and returns a String with the values of the instance variables. It should return a String formatted as:
"Name: <<Person's name>> Age: <<Person's age>> Employee Number: <<Employee Number>>"

- All must apply for full credit.

- Coding Style. (2pts)
  - Readable Code
    - Meaningful identifiers for data and methods.
    - Proper indentation that clearly identifies statements within the
      body of a class, a method, a branching statement, a loop statement,
      etc.
    - All the above must apply for full credit.

- Comments. (3pts)
  - Your name in the file. (1pts)
  - At least 5 meaningful comments in addition to your name. These must
    describe the function of the code it is near. (2pts)

Solution Tests:
- Does the solution compile?
- Does the solution have your name in the comments and 5 meaningful comments
  that describe the code?
- Does the output from the Question04 tester file match the following?

Does Employee inherit from Person?
Yes
Testing Employee's Default Constructor and toString Method
Name: none Age: 0 Employee Number: 0
Testing Employee's Param Constructor and toString Method
Name: asdf Age: 35 Employee Number: 22
Testing Employee's Param Constructor and toString Method with incorrect values
Name: asdf Age: 35 Employee Number: 0