**CSI 117 Review for the Final Exam**

## **Step 1:** Begin by reviewing the objectives listed at the start of each of the PPT lectures. If you do this for all the PPTs (before you begin this review), you will have a good foundation for the Final Review.

**Step 2:** Complete the final review.

## **Step 3**: Study the review you have created and practice problems similar to class and homework exercises.

|  |  |  |
| --- | --- | --- |
|  | **Definitions and Questions**  ***Find the definitions for the following terms in the Farrell***  ***text.*** | **Answer** |
| **Ex.** | **hardware** | **Hardware consists of the physical devices that make up a computer system.** |
| 1. | Structure Theorem |  |
| 2. | three valid control structures  used in structured design and programming |  |
| 3. | three types of selection  (identify the differences between the three types of selection) |  |
| 4. | two types of valid loops  (identify the differences between the two types of loops) |  |
| 5. | two types of pretest loops |  |
| 6. | three steps that must occur in  every loop |  |
| 7. | counter |  |
| 8. | accumulator |  |

|  |  |  |
| --- | --- | --- |
| 9. | sentinel |  |
| 10. | prime read |  |
| 11. | loop control variable |  |
| 12. | two common design and  programming mistakes often made when working with loops |  |
| 13. | How does a sentinel-controlled  loop work? |  |
| 14. | How does a flag-controlled  loop work? |  |
| 15. | variable |  |
| 16. | local variable |  |
| 17. | constant |  |
| 18. | class |  |
| 19. | object  After you find the definition, explain how to instantiate an object. |  |
| 20. | constructor |  |
| 21. | accessor |  |
| 22. | mutator |  |

|  |  |  |
| --- | --- | --- |
| 23. | file |  |
| 24. | method |  |
| 25. | module header  After you find the definition, list the elements that must be included in a module header. |  |
| 26. | module call  After you find the definition, list the elements that must be included in a module call. |  |
| 27. | argument  After you give the definition, tell what type of pseudocode statement would include an argument. |  |
| 28. | parameter  After you give the definition, tell what type of pseudocode statement would include a parameter. |  |
| 29. | pass by value |  |
| 30. | pass by reference |  |
| 31. | function header  After you find the definition, list the elements that must be included in a function header. |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 32. | function call  After you find the definition, list the elements that must be included in a function call. |  | | |
| 33. | Arguments and parameters  must agree in three ways. What are these? |  |  |  |
| 34. | UML |  |  |
| 35. | What is the primary purpose of  UML diagrams? |  |  |
| 36. |  |  |  |
| 37. |  |  |  |
| 38. |  |  |  |
| 39. | association relationship |  |  |
| 40. | multiplicity |  |  |
| 41. | private access  (After you give the definition, identify the elements of a class that are normally given private access.) |  |  |
| 42. | public access  (After you give the definition, identify the elements of a class that are normally given public access.) |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 43. | protected access  (After you give the definition, identify the elements of a class that may be given protected access and why this might be helpful.) |  |  |
| 44. | Identify and explain the five  features of object oriented analysis, design, and programming. Identify the advantages and disadvantages of each feature |  | Encapsulation  Information hiding Inheritance Polymorphism Data abstraction |
| 45. | Explain how a class differs  from an application. |  |  |

|  |  |
| --- | --- |
|  | |
|  | |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| **Write the data declaration statements for the following:** | |
| 50. a variable named pay | **Declare Real pay** |
| 51. a variable named count | **Declare Integer count** |
| 52. a variable named customerName | **Declare String customerName** |
| 53. a variable named middleInitial | **Declare Char middleInitial** |
| 54. a constant named MAX whose value is 100 | **Constant Integer MAX = 100** |
| 55. a House object named townhouse | **Declare House townhouse** |
| **Using the data declarations you wrote in the preceding problems, write the pseudocode instructions for the following**: | |
| 56. set the value of pay to 125.32 | **Set pay = 125.32** |
| 57. increment count by 1 | **Set count = count + 1** |
| 58. set the value of pay to whatever is in pay times MAX | **Set pay = pay \* MAX** |
| 59. set the customerName to Williams | **Set customerName = “Williams”** |
| 60. set the middleInitial to A | **Set middleInitial = ‘A’** |
| 61. display the value of customerName and pay | **Display customerName, pay** |

**Matching.** Place the **letter** of the calling statement next to the method header that matches.

Use the following data declarations: **Declare Integer numOne Declare Real numTwo Declare String stringOne Declare Boolean status Declare House myCityHouse Declare Shoe danceShoe**

|  |
| --- |
| **Calling Statement** |
| **A.** Call printInfo (numOne, status) |
| **B.** Call printInfo (danceShoe, myCityHouse) |
| **C.** Call printInfo (myCityHouse, danceShoe) |
| **D.** Call printInfo (numOne, numTwo, myCityHouse) |
| **E.** Call printInfo (myCityHouse, numOne, numTwo) |
| **F.** Call printInfo (numTwo, myCityHouse) |
| **G.** Set status = printInfo (stringOne) |
| **H.** Call printInfo (numOne, stringOne) |
| **I.** Call printInfo (numOne, danceShoe, status) |
| **J.** Set numOne = printInfo (stringOne) |
| **K.** Call printInfo (stringOne, numTwo) |

|  |  |  |
| --- | --- | --- |
| **Method Header** |  |  |
| Public Module printInfo (Integer a, String b) | 62. | **H** |
| Public Function Integer printInfo (String a) | 63. | **J** |
| Public Module printInfo (House a, Shoe b) | 64. | **C** |
| Public Function Boolean printInfo (String a) | 65. | **G** |
| Public Module printInfo (String a, Real b) | 66. | **K** |
| Public Module printInfo (Integer a, Boolean b) | 67. | **A** |
| Public Module printInfo (Real a, House b) | 68. | **F** |
| Public Module printInfo (Integer a, Real b, House c) | 69. | **D** |
| Public Module printInfo (House a, Integer b, Real c) | 70. | **E** |
| Public Module printInfo (Shoe a, House b) | 71. | **B** |
| Public Module printInfo (Integer a, Shoe b, Boolean c) | 72. | **I** |

|  |
| --- |
| **Vehicle** |
| **-price: Real**  **-model: String** |
| **+Vehicle()**  **+Vehicle(newPrice: Real, newModel: String)**  **+getPrice(): Real**  **+setPrice(newPrice: Real)**  **+getModel(): String**  **+setModel(newModel: String)** |

|  |  |
| --- | --- |
| **Assume a service class named Vehicle is being developed and includes attributes for price and model. Use the Vehicle class to answer the following questions.** | |
| 73. Write the pseudocode data declarations for the Vehicle  attributes. A sample price is 12599.99. A sample model is "Camry". | **Declare Real price Declare String model** |
| 74. Write the pseudocode for the default constructor. | **Public Module Vehicle() Set price = 0.0**  **Set model = “”**  **End Module** |
| 75. Write the pseudocode for another constructor that will  initialize the price and model to values specified when the object is instantiated. Be sure to incorporate adequate error checking. | **Public Module Vehicle (Real newPrice, String newModel) Set model = newModel**  **If NewPrice > 0 Then Set price = newPrice**  **Else**  **Display newPrice, “ is an invalid price” End If**  **End Module** |
| 76. Write the pseudocode for the accessor method getPrice ()  that will return the price of the vehicle. | **Public Function Real getPrice() Return price**  **End Function** |
| 77. Write the pseudocode for the accessor method getModel ()  that will return the model of the vehicle. | **Public Function String getModel() Return model**  **End Function** |
| 78. Write the pseudocode for the mutator method setPrice ()  that will set the price of the vehicle to the value it receives as a parameter. Be sure to incorporate adequate error checking. | **Public Module setPrice(Real newPrice) If newPrice > 0 Then**  **Set price = newPrice Else**  **Display newPrice, “ is an invalid price” End If**  **End Module** |

|  |  |
| --- | --- |
| 79. Write the pseudocode for the mutator method setModel ()  that will set the model of the vehicle to the value it receives as a parameter. | **Public Module setModel(String newModel) Set model = newModel**  **End Module** |
| 80. Write a statement that will instantiate an object of Vehicle  type named smithCar using the default constructor. | **Set smithCar = New Vehicle()** |
| 81. Write a statement that will instantiate an object of Vehicle  type named georgeTruck using the second constructor to initialize the price to 22000 and the model to "Dakota". | **Set georgeTruck = New Vehicle(22000, “Dakota”)** |
| 82. Write a statement to set the smithCar price to 2500. | **Call smithCar.setPrice(2500)** |
| 83. Write a statement to set the smithCar model to "Civic". | **Call smithCar.setModel(“Civic”)** |
| 84. Write a statement to display the price of the georgeTruck. | **Display georgeTruck.getPrice()** |
| 85. Write a statement to display the model of the georgeTruck. | **Display georgeTruck.getModel()** |
| 86.  part A:  Assume a sell () method will be placed in an application program. Write the method header for the sell () method  which receives a Vehicle object and customer name as  parameters.  (Hint: Think about the number of parameters it must receive and their data types.)  part B:  Call the method named sell () and pass the georgeTruck and customerName.  (Remember: georgeTruck and customerName were  declared in earlier problems. For this problem, you may assume that georgeTruck and customerName have been declared.) | **Method Header:**  **Module sell (Vehicle oneVehicle, String customerName)**  **Call Statement:**  **Call sell (georgeTruck, customerName)** |

|  |  |
| --- | --- |
| **when testVal is:** | **the final value of x will be:** |
| 1 | **9** |
| 2 | **6** |
| 3 | **6** |
| 5 | **7** |
| 6 | **9** |
| 9 | **8** |
| -5 | **9** |
| 10 | **9** |

|  |  |
| --- | --- |
| **when testVal is:** | **the final value of x will be:** |
| 1 | **6** |
| 2 | **6** |
| 3 | **6** |
| 5 | **7** |
| 6 | **7** |
| 9 | **8** |
| -5 | **100** |
| 10 | **100** |

|  |  |  |
| --- | --- | --- |
| **Show main memory and specify what is output by each program or program segment.** | | |
| 87. | Declare Integer testVal Declare Integer x = 5 Select testVal  Case 2:  Case 3:  Set x = 6 Case 5:  Set x = 7 Case 9:  Set x = 8 Default:  Set x = 9 End Select | Determine what the value of x will be after executing the pseudocode.  Do this for each of the following values of testVal. |
| 88. | Declare Integer testVal  Declare Integer x = 5 Select testVal  Case 1:  Case 2:  Case 3:  Set x = 6 Case 4:  Case 5:  Case 6:  Set x = 7 Case 7:  Case 8:  Case 9:  Set x = 8 Default:  Set x = 100 End Select | Determine what the value of x will be after executing the pseudocode.  Do this for each of the following values of testVal. |

|  |  |  |
| --- | --- | --- |
| 89. | Module main()  Declare Integer sum = 0 Declare Integer x = 4  While x <= 9  Set sum = sum + x Set x = x + 1  End While  Display "sum is ", sum, " and x is ", x End Module | **Sum is 39 and x is 10** |
| 90. | rewrite the above class using a for loop | **Module main()**  **Declare Integer sum = 0 Declare Integer x**  **For x = 4 To 9**  **Set sum = sum + x End For**  **Display “sum is “, sum, “ and x is “, x** |
| 91. | Module main()  Declare Integer sum = 0 Declare Integer x = 4  Do  Set sum = sum + x Set x = x + 1  While x != 9  Display "sum is ", sum, " and x is ", x End Module | **sum is 30 and x is 9** |

|  |  |  |
| --- | --- | --- |
| 92. | Module main()  Declare Integer j = 2 Declare Integer k = 5 Declare Integer n = 9 Declare Integer m  While j < k  Set m = 6 While m < n  Display "hello", j, " ", m Set m = m + 1  End While Set j = j + 1  End While End Module | **hello2 6**  **hello2 7**  **hello2 8**  **hello3 6**  **hello3 7**  **hello3 8**  **hello4 6**  **hello4 7**  **hello4 8** |
| 93. | Module main()  Declare Integer value = 30 Declare Boolean flag = True  If value >= 49 Then Display "Roger Rabbit" If flag == True Then  Display "Roger Ranger" Else  Display "Roger Riddle" End If  Else  Display "Roger Ringer" If flag == False Then  Display "Roger Rambler" Else  Display "Roger Roper" End If  End If End Module | **Roger Ringer Roger Roper** |

|  |  |  |
| --- | --- | --- |
| 94. | Module main()  Declare Integer value = 7 Declare Integer num = 10  Display "In main, before call" Display num, " ", value Call testMethod (15)  Display "In main, after call" Display num, " ", value  End Module  Module testMethod (Integer value) Declare Integer num = 25  Display "In testMethod" Display num, " ", value  End Module | **In main, before call 10 7**  **In testMethod 25 15**  **In main, after call 10 7** |

|  |  |  |
| --- | --- | --- |
| 95. |  |  |

1. Using the following pseudocode, add the appropriate statements to the program so that one of the following messages is printed based on the value of age:

### age message

greater than 100 "Wise one"

100 or less, but greater than or equal to 85 "Approaching wisdom"

less than 85 but greater than 65 "In training"

all others "Youth"

### Module main () Declare Integer age

**Display "Enter the age:" Input age**

**// put the rest of the pseudocode statements below**

### If age <= 65 Then Display “Youth”

**Else If age <= 84 Then Display “In training”**

**Else If age <= 100 Then**

**Display “Approaching wisdom” Else**

**Display “Wise one” End If**

**If age >= 101 Then Display “Wise one”**

**Else If age >= 85 Then**

**Display “Approaching wisdom” Else If age >= 66 Then**

**Display “In training”**

**Else**

**Display “Youth” End If**

**End Module**

1. Using the following pseudocode, add the appropriate statements to the program so that the variable product contains the product of the odd numbers between 3 and 11 inclusive, using a While loop

### Module main ()

**Declare Integer product Declare Integer num**

**// put the rest of the pseudocode statements below**

### Set product = 1 Set num = 3 While num <= 11

**Set product = product \* num Set num = num + 2**

**End While**

**Display "The product of the odd numbers between 3 and 11 inclusive is", product End Module**

## Draw a UML class diagram for the program that contains the following classes, showing all the relationships and including multiplicity:

* 1. A service class named Home stores the type of home (single-family, townhouse, condo, etc.), the age of the home, and the tax value of the home. Include constructors, accessors, and mutators. Include a method that prints all the information stored about the home.
  2. A sevice class named Owner that stores the owner’s name and a Home object. Include constructors, accessors, and mutators. Include a method that prints all the information about an owner.
  3. A HouseAddress service class that stores a street address, the zip code, and an Owner object. Include constructors, accessors, and mutators. Include a method that prints all the information about a house address
  4. An application program with a main() method that declares and uses exactly 3 HouseAddress objects.

1. Develop a program that calculates an employee’s average yearly salary (ays). Create the following:

* a class diagram which includes any relationships and multiplicity and
* the pseudocode.
  1. An Employee service class that stores the following:
     + employee id number (such as "PR1296"),
     + the number of years the employee has worked for the company (such as 12), and
     + the employee’s ays (such as 22599.99).

Include a default constructor. Include a second constructor that receives two parameters—the first should be the employee id number and the second should be the number of years employment. Include accessor and mutator methods for the employee id

number and the number of years the employee has worked for the company, but not for the ays. **Be sure to incorporate adequate error checking for these methods.** Include an accessor method for ays. Create a method named calculateAys() that

calculates and prints the ays. The method inputs the salary the employee has earned each year, which the user will enter on the keyboard. The calculateAys() method should input one salary for each year the employee has worked for the company. The ays is calculated by adding up all the yearly salaries and dividing the total by the number of years the employee has worked for the

company.

* 1. An application program that contains the main method. The main program creates an employee object named panPeter and initializes this object using the default constructor, The main program creates another employee object named ozDorothy using

the second constructor to initialize her employee id number to "OZ1234" and her years of employment to 5, The program

should then set the employee id for panPeter to "PA9876" and the number of years of employement for panPeter to 7. For each object, the program should display the employee id number, display the number of years the employee has worked for the company, and calculate and display the employee’s ays, calling methods in the Employee class, as appropriate.

# Class Employee Private String idNum

**Private Integer yrsWorked Private Real ays**

**Public Module Employee() Set idNum = “ “**

**Set yrsWorked = 0 Set ays = 0.0**

**End Module**

**Public Module Employee(String newIdNum, Integer newYrsWorked) Set idNum = newIdNum**

**Call setYrsWorked (newYrsWorked)**

**Set ays = 0.0 End Module**

**Public Function String getIdNum() Return idNum**

**End Function**

**Public Module setIdnum(String newIdNum) Set idNum = newIdNum**

**End Module**

**Public Function Integer getYrsWorked() Return yrsWorked**

**End Function**

**Public Module setYrsWorked (Integer newYrsWorked) If newYrsWorked >= 0 AND newYrsWorked <= 70**

**Set yrsWorked = newYrsWorked Else**

**Set yrsWorked = 0**

**Display newYrsWorked, “ is invalid; it must be between 0 and 70” End If**

**End Module**

**Public Function Real getAys() Return ays**

**End Function**

**Public Module calculateAys() Declare Real salary**

**Declare Real totalSalaries = 0.0 Declare Integer year = 0**

**For year = 1 To yrsWorked**

**Display “Enter the salary for year “, year**

**Input salary**

**Set totalSalaries = salary + totalSalaries End For**

**Set ays = totalSalaries / yrsWorked**

**Display “The average yearly salary is “, ays End Module**

**End Class**

**Module main()**

**Declare Employee panPeter Set panPeter = New Employee() Declare Employee ozDorothy**

**Set ozDorothy = New Employee(“OZ1234”, 5) Call panPeter.setIdNum(“PA9876”)**

**Call panPeter.setYrsWorked(7)**

**Display “Id number for panPeter is “, panPeter.getIdNum()**

**Display “panPeter has worked here for “, panPeter.getYrsWorked(), “ years” Call panPeter.calculateAys()**

**Display “The average yearly salary for panPeter is “, panPeter.getAys() Display “Id number for ozDorothy is “,ozDorothy.getIdNum()**

**Display “ozDorothy has worked here for “,ozDorothy.getYrsWorked(), “ years” Call ozDorothy.calculateAys()**

**Display “The average yearly salary for ozDorothy is “,ozDorothy.getAys() End Module**