David Martinez

Professor William Crosbie

Object Oriented Programming

04 October 2015

2.12) The type for each field is int, Student and Server.

2.13) The names for each field are alive, tutor and game.

2.14) Student, Server, Person and Game are all class names.

2.17) private int status;

2.21)

public Pet(String petsName)

{

name = petsName;

}

2.22) public Date(String newMonth, int newDate, int newYear)

2.24) I would characterize as “How much money do I have?”

2.26)

public class TicketMachine

{

private int total;

private int price;

public int getPrice()

{

return price;

}

public int getTotal()

{

return total;

}

}

2.27) “Missing return statement” is the error upon compilation.

2.29) These methods do not have return statements. These methods do not have to return any value, but simply perform a function. The keyword “void” in the headers would indicate the fact that they do not require return statements.

2.44)

public TicketMachine(int newPrice)

{

price = new Price;

}

public TicketMachine()

{

price = 8675309;

}

2.47) If the operator is changed to a greater than or equal to operator, I predict that it will allow the number zero to be entered as a valid amount to add to one’s balance. This is indeed true upon testing. No other major ramifications occurred.

2.49) The feature controlled by a Boolean value was visibility. This was well-suited to being controlled by a Boolean value, as it made sense when the object was asking whether or not the object was indeed visible.

2.50) Codes 2.1 and 2.8 differ in that 2.8 checks variables and input to make sure they are relevant and functional data for the program at hand.

2.51) It is possible to do this, but if one’s balance is too low, the method simply does not do anything, making the program less robust overall.

2.52) No, a negative value could never be printed as a price greater than the balance could never be subtract from the balance.

2.54) int saving = price \* discount;

2.55) int mean = total/count; //This answer makes me uncomfortable because it is my instinct to either typecast or change variable types here for the sake of accuracy.

2.56)

if (price > budget)

{

System.out.println(“Too expensive);

} else {

System.out.println(“Just right);

}

2.60) This constructor creates local variable which it intends to use globally. This class should have price as a class variable.

2.61)

public int emptyMachine()

{

int machineTotal = total;

total = 0;

return machineTotal;

}

2.64)

Name: getCode

Return type: String

2.65)

Method name: setCredits

Parameter name: creditValue

Parameter type: int

2.69)

public Person(String myName, int myAge)

{

name = myName;

age = myAge;

}

2.71)

public String getName()

{

return name;

}

2.72)

public void setAge(int newAge)

{

age = newAge;

}

2.80) This returns 1500 both times.

2.81) t2 should return 1500 because it uses the value from t1.

2.82) It should return 2000 as t2 is simply a reference to t1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| !(A && B) | 0 | 1 | (A || B) | 0 | 1 |
| 0 | False | True | 0 | False | True |
| 1 | True | False | 1 | True | True |

These statements are not equal. It said change the second term of “the Boolean statement” and I wasn’t sure which one. If the second one were just an XOR (^) then the two statements would be equivalent.

3.18) The numbers zero through four are possible outcomes.

3.19) The numbers zero though the upper limit of 32-bit integers less one would be possible outcomes.

3.20) The increment method increments the value by one. Then the modulus is used to compare the incremented value to the limit. If the two are the same, the modulus will return 0. This resets the value.

3.21)

public void increment()

{

value++;

if (value == limit)

{

value = 0;

}

}

This is certainly less efficient than by using the modulus method.

3.23) The clock starts at 00:00 because NumberDisplay classes are initialized to zero.

3.26) public Editor(String newFilename, int newPosition)

3.28) The second constructor sets the time of the clock with two integers. It sets the numbers in NumberDisplay using its constructor and the integers.

3.29) The setTime method includes an updateDisplay method, making that statement individually unnecessary.

3.30) print(“doge.txt”, false); print(“MFDOOM.txt”, true); getStatus(9001); getStatus(42);

3.35-3.37) Screenshots taken.

3.38) I predict the else block to be executed next, given that the if statement is not satisfied. I was correct for the reason previously mentioned.

3.39) This time, it actually hits the null condition and follows through with the action.

3.40) On this occasion, I am brought to the print method inside the MailItem class.

3.41) Screenshot taken.

3.42)

MailClient

get()

print()

send(to, message)

nextMailItem

newMailItem

This diagram is strictly in an effort to show object interaction. The “nextMailItem” must already be extant in an effort to be used. In order for it to be printed, it must first be gotten, which is something that the get method can do independently as well. The get method returns the object of type mailItem. The sendMailItem generates a new object of mailItem type in order to send it.