Eric Wallace

CIS225

Assignment1.2

2.2- The balance is 0.

2.3- I didn’t see where a refund was issued, it just cleared to a 0 balance.

2.4- Did it.

2.5- It looks the same but with the different price displayed for the ticket value.

2.6-

public class Student

{

}

public class LabClass

{

}

2.7- Did it, there were several errors and it would not compile. Yes and the errors are cleared.

2.8- Did it and it compiled without error.

2.9- Taking class out returns errors when compiled.

2.10- fields: price, balance, total

Constructor:

public TicketMachine(int cost)

Methods:

public int getPrice()

public int getBalance()

public void insertMoney(int amount)

public void printTicket()

2.11- the fields are private where the constructors are public. The constructors are used to alter the original parameters into the fields. Methods can then manipulate the data entered to change the object.

2.12- integer, string, string

2.13- alive, tutor, game

2.14- boolean, Person, Game

2.15- There were errors with every edit of the declaration until I returned it back to normal.

2.16- Removing the semi-colon breaks the program.

2.18- Student class.

2.19- Two, String title is a string and double price is an integer to the second decimal.

2.20- Chapters, Sections, Pages, Paragraphs, Diagrams

2.21-

public Pet(String petsName)

{

name = petsName;

}

Public getName()

{

return name;

}

2.22-

public Date(int month, int date, int year)

2.23- I don’t see a difference between them.

2.24- “What is my current balance?”

2.25- No the return statement did not have to be changed. The method takes precedence over the field.

2.27- Missing return statement.

2.28- The printTicket method is performing a calculation before resetting the balance at the end. The getPrice is just returning the amount.

2.29- No they do not have return statements. They have void in the header which specifies that the method does not return a result.

2.31- The method is performing the action of setting the price and does not return an object.

2.33-

public void increase(int points)

{

score += points;

}

2.34- Yes as the score increases the method mutates it to add the parameter set by points.

2.35-

Public void discount(int amount)

{

price = price – amount;

}

2.36- My cat has green eyes.

2.38- It prints the string # price cents.

2.39- Still not printing properly.

2.40- No. Neither option provided a proper price on the tickets.

2.42- Both show different outputs respective to their entered price. It is calling the method parameter set in each and displaying the same information but for the different machines.

2.43- The ticket-machine compiles and instead of asking what the cost will be, it just prompts for a name. After compiling, when the getPrice method is called it prompts the price at 1000.

2.45- Done, no parameters. It worked. The empty method is a mutator.

2.46- The balance doesn't change when an error prints. Error will print because the parameter must be greater than zero, not greater than or equal to zero.

2.47- If the operator is changed it triggers a print to request using a number greater than zero.

2.49- The boolean field determined whether it was visible or not.

2.50- 2.8 returns a calculation with accurate balance amount. 2.1 just clears the balance.

2.51- Nothing happened when I tried printing without inserting money.

2.52- No the if-statement should prevent that from happening as it will only print the ticket when the balance is greater than or equal to the price of the ticket, triggering the else statement if it isn't.

2.53- Done.

2.54- saving = price \* discount

2.55- mean = total / count

2.56-

if (price > budget) {

System.out.println("Too expensive");

}

else {

System.out.println("Just right");

}

2.57-

if (price > budget) {

System.out.println("Too expensive for your" + budget + "budget.");

}

else {

System.out.println("Just right");

}

2.58- Because it is just setting the balance to zero after the ticket is printed.

2.59- Error, unreachable statement. The return ends the compiling before the balance is set.

2.60- The int added to price in the method. It compiles but, doesn't list a price.

2.61-

2.63-

2.64- Name is getCode, the type is a string.

2.65- Name is setCredits, parameter name is creditValue and it is an integer.

2.66-

public class Person

{

}

2.67-

private String name;

private int age;

private String code;

private int credits;

2.68-

public Module (String moduleCode)

{

code = moduleCode;

}

2.69-

public Person(String myName, int myAge)

{

name = myName;

age = myAge

}

2.70-

public int getAge()

{

return age;

}

2.71-

public String getName()

{

return name;

}

2.72-

public setAge(int age)

{

age = setAge;

}

2.73-

public void printDetails()

{

if True{

System.out.println("The name of this person is" + name);

}

}

2.75- "Henr557"

2.76- An error is presented because their was no validation on the input. If it was a valid input then it the method would need to be updated to have less than the first four letters in the first name.

2.79- yes, in Java an operator can concatenate with an integer and also add integer's together

2.80- It updated to what money was inserted.

2.81- Same as what t1 had for a balance.

2.82- We set t2 to equal t1 so it appears that whatever is changed will translate to other copies made from the original.

2.86- Yes, because the code as it stands just returns the value unchanged.

2.88- Done.

2.89- Done.