PASS SWOT Revision - Sample Questions

Question 1

Assume fiction is an object of Book class and the method getFee() returns the value of the loan fee. These have defined elsewhere. If the current loan fee is 8 dollars. What would be the value of the variable **amount** after the evaluation of the following piece of Java code?

```
int amount = fiction.getFee()

amount+1;

amount *= amount++;
```

81

Question 2

How many times will the body of the following loop be executed? Assume the method getDeposit() has been defined elsewhere.

```
for (int count = 0; count < 15; count++)

{

        balance = balance + getDeposit();

        count + 1;

}
```

16 without count + 1; is 15

Question 3

After the following declaration:

People[] crowd = new People[10];

How many People objects are there? Explain your answer.

Size is 10

List is null so zero object

Question 4

State the overall purpose of the following segment of code. Assume that a,b, and c are declared to be of type int and have been initialized. Note that we are looking for the overall goal of the code, not explanation of each individual line. This should be one short sentence.

```
c = a;

a = b;

b = c;
```

Swap a and b value

Question 5

What is the difference between testing and debugging?

Test just process bug

Debug correct bug

Question 6

This semester we discussed responsibility-driven design. Explain how responsibility-driven design could have been used in the process of designing the program for the second assignment “Where is Migaloo”. How does responsibility-driven design improve program design?

Question 7

Write a constructor for a class called **ClubMember**. The constructor has three parameters. The first is of type String and is called **name**. The second is of type int and is called **age**. The third is of type double and is called **membershipFee**. The name should not be blank, the age should be greater or equal to 18 and the membershipFee is either $50 or $100. If any value is invalid then the constructor set the fields to sensible values.

```java
public class ClubMember
{
    private String name;
    private int age;
    private double membershipFee;

    public ClubMember()
    {
        name = "";
        age = 0;
        membershipFee = 0;
    }

    public ClubMember(String name, int age, double membershipFee)
    {
        if(!name.length().isBlank)
            this.name = name;
        else
            name = "Error"
        if(age >=18)
            this.age = age;
        if(membershipFee = 50 || membershipFee = 100)
            this.membershipFee = membershipFee;
    }
}
```

Question 8

Write the complete Java code for a method taht will read a series of temperatures from the keyboard, then display on the screen the **number of temperatures** over 40 degree, and the **average** of all the temperatures. The method will terminate when a temperature of -99 is entered. A temperature of -99 is not to be regarded as a valid value.

Your method should work for any number of temperatures entered.

```java
public void countAverage()
{
    Scanner console = new Scanner(System.in);
    System.out.println("Please input a temperatures:");
    temperature = console.nextInt();
    int total = 0;
    int count = 0;‘
    int average = 0;
    while(!temperature = -99 && temperature > 40)
    {
        total += temperature;
        average = total/count;
        System.out.println("The temperature is: " + temperature);
        System.out.println("The average temperature is: " + average);
    }
}
```

Question 9

Write a method called displayOrders that will write the details of a fast food company’s orders to the screen.

These details are stored in objects of the class oOrder and each object of class Order is stored in an ArrayList called **orders**. You may assume that orders has been declared elsewhere in the class containing the method **displayOrders** and is visible inside **displayOrders**.

An **Order** has many attributes, but the only details of an order to be written to screen are the following, which are stored as individual fields inside an **Order** object.

- **orderNumber**, of type int, that uniquely denotes the order.
- **customerName**, of type String.
- **suburb**, of type String indicating the suburb the order was sent to.
- **totalPrice**, of type double showing the total price of all items in an order

The method **displayOrders** takes no parameters and returns nothing.

The screen will show the details of each order on separate lines in the form:

1001 John Citizen Caulfield 24.00

You may assume class **Order** has all the required **get/set** methods for all included attributes.

```
public class oOrder
{
    private ArrayList<Order> orders;

    public void displayOrders()
    {
        for(Order order : orders)
        {
            System.out.println(order.getOrderNumber());
        }
    }
}
```