EECS 1022 3.0 Programming for Mobile Computing

Solution of Midterm - Version B

18:30–19:30 on July 10, 2017

1 (2 marks)

(a) The process of software development consists of multiple phases. Which artefact is produced in the implementation phase?

Answer: code.

Marking scheme: 1 mark for code.

(b) Which phase is before the implementation phase?

Answer: design.

Marking scheme: 1 mark for design.

2 (2 marks)

The design pattern MVC separates the code into a model, a view, and a controller. Assume you develop a mobile app that asks the user to enter a number in a text box and makes the Android device vibrate if that number is prime. For each of the following items, indicate whether it belongs to the model, the view or the controller.

(a) The text box to enter the number.

Answer: view.

Marking scheme: 0.5 mark for view.

(b) The code that makes the Android device vibrate.

Answer: controller or activity.

Marking scheme: 0.5 mark for controller or activity.

(c) The code that tests whether the entered number is prime.

Answer: model.

Marking scheme: 0.5 mark for model.

(d) The code that extracts the number from the text box.

Answer: controller or activity.

Marking scheme: 0.5 mark for either controller or activity.

3 (2 marks)

Consider the API of the class `Student` which is provided at the end of this test.

(a) What is the return type of the `getNumber` method?

Answer: long.

Marking scheme: 1 mark for long.

(b) How many explicit parameters does the `equals` method have?

Answer: 1.

Marking scheme: 1 mark for 1.

4 (2 marks)

Can two objects have the same identity but different states? **Explain your answer.** You only get marks for your explanation.

Answer: No. If two objects have the same identity, then they reside on the same memory address, that is, they are one and the same object. Hence, their attributes have the same values and, therefore, they have the same state.

Marking scheme: 1 mark for mentioning that these objects reside on the same *memory address*. 1 mark for mentioning that their *attributes* have the same *values*.

5 (2 marks)

(a) Consider the following code snippet.

```
Student john = new Student("John Doe", 100001475, 3.26);
Student jane = new Student("Jane Doe", 100001476, 3.78);
john.setGpa(3.32);
```

Draw the corresponding memory diagram. Make sure that the attributes `name`, `number` and `gpa` of the `Student` class and the variables `john` and `jane` are reflected in your diagram.

Answer:

0		
⋮		
100	*300*	john
	400	jane
⋮		
200	Student class	
⋮		
300	Student object	
	”John Doe”	name
	100001475	number
	3.32	gpa
⋮		
400	Student object	
	”Jane Doe”	name
	100001476	number
	3.78	gpa
⋮		

”John Doe” and ”Jane Doe” are String objects and, hence, should be represented by object blocks as well, but for simplicity we have not done that here.

Marking scheme:

- 0.25 mark for two Student object blocks.
- 0.25 mark for the values of john and jane are the addresses of the two objects.
- 0.25 mark for the value of john’s gpa attribute being 3.32.
- 0.25 mark for the correct values of the attributes of the Student objects (apart from the gpa attribute of john)

(b) The method `equals` is implemented as follows.

```
public boolean equals(Object that)
{
  return this.number == ((Student) that).number;
}
```

Consider the following code snippet.

```
Student john = new Student("John Doe", 100001475, 3.26);
Student jane = new Student("Jane Doe", 100001476, 3.78);
boolean same = john.equals(jane);
```

Draw the corresponding memory diagram. Make sure that the attributes `name`, `number` and `gpa` and the variables `john`, `jane` and `same` are reflected in your diagram. Make sure to include the invocation block for the call of the `equals` method.

Answer:

0		
⋮		
100	*300*	john
	400	jane
	false	same
⋮		
200	Student class	
⋮		
300	Student object	
	”John Doe”	name
	100001475	number
	3.32	gpa
⋮		
400	Student object	
	”Jane Doe”	name
	100001476	number
	3.78	gpa
⋮		
500	equals invocation	
	300	this
	400	that
⋮		

”John”, ”Jane” and ”Doe” are String objects and, hence, should be represented by object blocks as well, but for simplicity we have not done that here.

Marking scheme:

- 0.2 mark for this in the invocation block.
- 0.2 mark for that in the invocation block.
- 0.2 mark for the value of this (address of the Student object referred to by john) in the invocation block

- 0.2 mark for the value of that (address of the Student object referred to by jane) in the invocation block
- 0.2 mark for the value of same.