EECS 1022 3.0 Programming for Mobile Computing

Solution of Midterm - Version C

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 analysis phase?

Answer: specification.

Marking scheme: 1 mark for specification.

(b) Which phase is after the analysis 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 translates a text, entered into a text box, into pirate slang and displays the result after the button is pressed. For each of the following items, indicate whether it belongs to the model, the view or the controller.

(a) The text box.

Answer: view.

Marking scheme: 0.5 mark for view.

(b) The code that extracts the entered text from the text box.

Answer: controller or activity.

Marking scheme: 0.5 mark for controller or activity.

(c) The code that translates the entered text into pirate slang.

Answer: model.

Marking scheme: 0.5 mark for model.

(d) The translate button.

Answer: view.

Marking scheme: 0.5 mark for view.

3 (2 marks)

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

(a) How many (public) methods does the `MemoryStick` class contain?

Answer: 7.

Marking scheme: 1 mark for 7.

(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*.

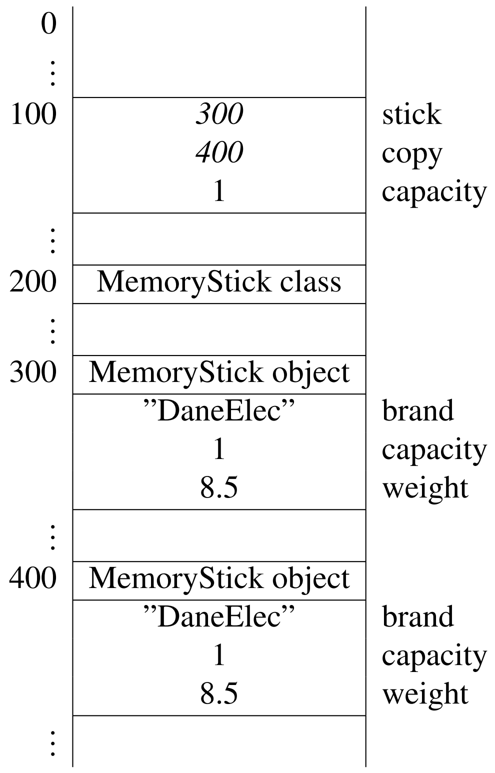
5 (2 marks)

(a) Consider the following code snippet.

```
MemoryStick stick = new MemoryStick("DaneElec", 1, 8.5);
MemoryStick copy = new MemoryStick(stick);
int capacity = stick.getCapacity();
```

Draw the corresponding memory diagram. Make sure that the attributes `brand`, `capacity` and `weight` of the `MemoryStick` class and the variables `stick`, `copy` and `capacity` are reflected in your diagram.

Answer:



"DaneElec" is a String object and, hence, should be represented by an object block as well, but for simplicity we have not done that here.

Marking scheme:

- 0.25 mark for two MemoryStick object blocks.
- 0.25 mark for the values of stick and copy are the addresses of the two objects.
- 0.25 mark for the value of capacity variable.
- 0.25 mark for the correct values of the attributes of the MemoryStick objects.

(b) The method is implemented as follows.

```
public int compareTo(MemoryStick that)
{
  return this.capacity - that.capacity;
}
```

Consider the following code snippet.

```
MemoryStick stick = new MemoryStick("DaneElec", 1, 8.5);
MemoryStick copy = new MemoryStick(stick);
int comparison = stick.compareTo(copy);
```

Draw the corresponding memory diagram. Make sure that the attributes `brand`, `capacity` and `weight` and the variables `stick`, `copy` and `comparison` are reflected in your diagram. Make sure to include the invocation block for the call of the `compareTo` method.

Answer:

0		
⋮		
100	*300*	stick
	400	copy
	0	comparison
⋮		
200	MemoryStick class	
⋮		
300	MemoryStick object	
	”DaneElec”	brand
	1	capacity
	8.5	weight
⋮		
400	MemoryStick object	
	”DaneElec”	brand
	1	capacity
	8.5	weight
⋮		
500	compareTo invocation	
	300	this
	400	that
⋮		

”DaneElec” is a String object and, hence, should be represented by an object block 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 MemoryStick object referred to by stick) in the invocation block

- 0.2 mark for the value of that (address of the MemoryStick object referred to by copy) in the invocation block
- 0.2 mark for the value of comparison.