Week 9 Pre-reading quiz

Test your knowledge for Week 9!

Question 1 *Submitted May 11th 2022 at 5:15:24 pm*

Consider the following code fragment:

Which of the following is NOT true about this code fragment?

- Person is a class name
- setAge is a Mutator method
- Using the 'new' keyword to create a new instance is optional
- 21 is a parameter
- tania identifies (references) a particular Person instance
- If you think all of the above are true, select this option

Question 2 *Submitted May 11th 2022 at 5:15:50 pm*

Which of the following is NOT true?

- Accessor methods must include a return statement with a return expression
- Mutator methods must have an input parameter
- Mutator method code is responsible for maintaining the integrity of an object's data

- [ ] Accessor and Mutator methods are always public because they are intended to be used by code outside their class
- [x] Mutator methods must include a return statement with a return expression
- [ ] If you think all of the above are true, select this option

Question 3 *Submitted May 11th 2022 at 5:16:04 pm*

Where is the guardian code in the **Date** class located?

- [ ] In a mutator
- [ ] In an accessor
- [ ] In a constructor
- [x] In a helper method (i.e. a method that helps some other (usually public) method perform its task)
- [ ] If you think none of the above are the correct location, select this option

Question 4 *Submitted May 11th 2022 at 5:16:12 pm*

Why are **Date** and **Food** objects immutable?

- [ ] They only have 1 constructor
- [ ] Their constructors are full parameter constructors
- [x] They have no mutators
- [ ] Their instance variables are primitive values

- [ ] If you think none of the above are the reason why objects of these classes are immutable, select this option

Question 5 *Submitted May 11th 2022 at 5:16:44 pm*

Which of the following is false about the **validateDate** method in the **Date** class?

You can assume it has been fully coded (it's currently little more than a stub method).

- [ ] It could be public without compromising data hiding in the Date class
- [ ] It could be static without causing a compile error
- [ ] It could be static without causing a logic error during use
- [x] If you think none of the above are false, select this option

Question 6 *Submitted May 11th 2022 at 5:16:53 pm*

Consider the **Monster** class's two constructors. Which of the following is false?

- [ ] They both access a static variable
- [ ] The parameters of both do not initialise all instance variable using incoming parameters
- [x] Neither uses any mutators to protect any instance variable values
- [ ] If you think none of the above are false, select this option

Question 7 *Submitted May 11th 2022 at 5:17:12 pm*

Does the variable **numMonstersInstantiated** in the **Monster** class have to be static to perform its function?

- [x] Definitely Yes, no caveats

- Definitely No, no caveats
- Static or non-static will work but good coding style requires static
- Static or non-static will work but good coding style requires non-static

Question 8 *Submitted May 11th 2022 at 5:17:22 pm*

Does the method **getNumMonstersInstantiated** in the **Monster** class have to be static to perform its function?

- Definitely yes, no caveats
- Definitely no, no caveats
- **Static or non-static will work but good coding style requires static**
- Static or non-static will work but good coding style requires non-static

Question 9 *Submitted May 11th 2022 at 5:18:00 pm*

Which of the following **Zoo** methods will not be adversely affected if the variable **numberOfMonsters** was public and its integrity compromised by driver code?

- getIndexOfAMonster
- getAMonsterByIndex
- getAMonsterByName
- addAMonster
- insertAMonster
- toString

- [x] If you think all of the above will be adversely affected, select this option

Class As Reference Types: Scenario 1

T2 – SCENARIO 1: Line 53 changed

```
39  // REFERENCE TYPES
40  Person p1, p2;
41
42  System.out.println("\n >>> Reference Types");
43  p1 = new Person();  // instantiation a Person object
44  p2 = new Person();  // instantiation another Person object
45  System.out.print("p1 name/age = " + p1.getName() + "/" + p1.getAge());
46  System.out.println(", p2 name/age = " + p2.getName() + "/" + p2.getAge());
47
48  p1.setName("Alana"); p1.setAge(1);
49  p2.setName("Jake"); p2.setAge(2);
50  System.out.print("p1 name/age = " + p1.getName() + "/" + p1.getAge());
51  System.out.println(", p2 name/age = " + p2.getName() + "/" + p2.getAge());
52
53  p2.setName(p1.getName()); p2.setAge(p1.getAge());
54  // BUT p1 & p2 are interpreted as memory address
55  System.out.print("p1 name/age = " + p1.getName() + "/" + p1.getAge());
56  System.out.println(", p2 name/age = " + p2.getName() + "/" + p2.getAge());
57
58  p1.setAge(7);
59  System.out.print("p1 name/age = " + p1.getName() + "/" + p1.getAge());
60  System.out.println(", p2 name/age = " + p2.getName() + "/" + p2.getAge());
```

Question *Submitted Jun 10th 2022 at 11:52:22 am*

Scenario 1: What is the output of the display statements?

Line 45 & 46: p1 name/age =___/___, p2 p1 name/age = ___/___

Line 50 & 51: p1 name/age =___/___, p2 p1 name/age = ___/___

Line 55 & 56: p1 name/age =___/___, p2 p1 name/age = ___/___

Line 59 & 60: p1 name/age =___/___, p2 p1 name/age = ___/___

Re-order the items below to get correct order of displayed outputs.

p1 name/age = null/0, p2 name/age = null/0

p1 name/age = Alana/1, p2 name/age = Jake/2

p1 name/age = Alana/1, p2 name/age = Alana/1

p1 name/age = Alana/7, p2 name/age = Alana/1

Class As Reference Types: Scenario 2

T2 – SCENARIO 2: Line 53 & 58 changed

```
39  // REFERENCE TYPES
40  Person p1, p2;
41
42  System.out.println("\n >>> Reference Types");
43  p1 = new Person();  // instantiation a Person object
44  p2 = new Person();  // instantiation another Person object
45  System.out.print("p1 name/age = " + p1.getName() + "/" + p1.getAge());
46  System.out.println(", p2 name/age = " + p2.getName() + "/" + p2.getAge());
47
48  p1.setName("Alana"); p1.setAge(1);
49  p2.setName("Jake"); p2.setAge(2);
50  System.out.print("p1 name/age = " + p1.getName() + "/" + p1.getAge());
51  System.out.println(", p2 name/age = " + p2.getName() + "/" + p2.getAge());
52
53  p2.setName(p1.getName()); p2.setAge(p1.getAge());
54  // BUT p1 & p2 are interpreted as memory address
55  System.out.print("p1 name/age = " + p1.getName() + "/" + p1.getAge());
56  System.out.println(", p2 name/age = " + p2.getName() + "/" + p2.getAge());
57
58   p1.setName("test"); p1.setAge(7);
59  System.out.print("p1 name/age = " + p1.getName() + "/" + p1.getAge());
60  System.out.println(", p2 name/age = " + p2.getName() + "/" + p2.getAge());
```

Question *Submitted Jun 10th 2022 at 11:52:43 am*

Scenario 2: What is the output of the display statements?

Line 45 & 46: p1 name/age =___/__, p2 p1 name/age = ___/___

Line 50 & 51: p1 name/age =___/__, p2 p1 name/age = ___/___

Line 55 & 56: p1 name/age =___/__, p2 p1 name/age = ___/___

Line 59 & 60: p1 name/age =___/__, p2 p1 name/age = ___/___

Re-order the items below to get correct order of displayed outputs.

p1 name/age = null/0, p2 name/age = null/0

p1 name/age = Alana/1, p2 name/age = Jake/2

p1 name/age = Alana/1, p2 name/age = Alana/1

p1 name/age = test/7, p2 name/age = Alana/1

Class As Reference Types: Scenario 3

T2 – SCENARIO 3: Line 53 unchanged & 58 changed

```
39  // REFERENCE TYPES
40  Person p1, p2;
41
42  System.out.println("\n >>> Reference Types");
43  p1 = new Person();  // instantiation a Person object
44  p2 = new Person();  // instantiation another Person object
45  System.out.print("p1 name/age = " + p1.getName() + "/" + p1.getAge());
46  System.out.println(", p2 name/age = " + p2.getName() + "/" + p2.getAge());
47
48  p1.setName("Alana"); p1.setAge(1);
49  p2.setName("Jake"); p2.setAge(2);
50  System.out.print("p1 name/age = " + p1.getName() + "/" + p1.getAge());
51  System.out.println(", p2 name/age = " + p2.getName() + "/" + p2.getAge());
52
53  p1 = p2;    // p2 is set with a copy of the value of p1
54  // BUT p1 & p2 are interpreted as memory address
55  System.out.print("p1 name/age = " + p1.getName() + "/" + p1.getAge());
56  System.out.println(", p2 name/age = " + p2.getName() + "/" + p2.getAge());
57
58   p1.setName("test"); p1.setAge(7);
59  System.out.print("p1 name/age = " + p1.getName() + "/" + p1.getAge());
60  System.out.println(", p2 name/age = " + p2.getName() + "/" + p2.getAge());
```

Question *Submitted Jun 10th 2022 at 11:53:04 am*

Scenario 3: What is the output of the display statements?

Line 45 & 46: p1 name/age =___/___, p2 p1 name/age = ___/___

Line 50 & 51: p1 name/age =___/___, p2 p1 name/age = ___/___

Line 55 & 56: p1 name/age =___/___, p2 p1 name/age = ___/___

Line 59 & 60: p1 name/age =___/___, p2 p1 name/age = ___/___

Re-order the items below to get correct order of displayed outputs.

p1 name/age = null/0, p2 name/age = null/0

p1 name/age = Alana/1, p2 name/age = Jake/2

p1 name/age = Alana/1, p2 name/age = Alana/1

p1 name/age = test/7, p2 name/age = test/7

Code Prac: Multi-Class App

You are required to write a Java program that will display the time. Keep in mind that to facilitate OOP type design, you must write your code so they are reusable. Thus for this program, you must appropriate the following design:

You are required to follow the above guidelines when designing your program. For now, you do not need to do validations to check if the time is correct. The objective is to ensure you understand a multiple object creation, working with multiple classes, and invoking methods using objects.

Hint: You do not need conditional statements right now—this is something we will learn in upcoming lessons. The program can be done without these. Also remember the modulus (%) operator, it might be useful here.