Testing I Quiz

TOTAL POINTS 10

1. Which of the following are reasons to write tests (select all that apply)? 1 point

- [x] to ensure that new functionality operates as expected
- [ ] to make apps run faster by reducing binary sizes
- [ ] to measure how effective software developers are
- [x] to help make sure that changes to code do not break existing functionality

2. When building a new app, which of the following are true (select all that apply)? 1 point

- [ ] There is exactly one way to implement it.
- [x] Measuring the impact of changes is critical.
- [ ] There is exactly one way to design it.
- [x] Finding the right design may be an iterative process.

3. What is a unit test (select all that apply)? 1 point

- [ ] a test that measures performance
- [x] a test that isolates a specific component for evaluation
- [ ] a test of a single user interface element
- [ ] a test that isolates multiple components for evaluation
- [ ] a measurement of test coverage

4. Which Java framework is used to write unit tests in Android (select all that apply)? 1 point

- [ ] Dagger
- [ ] Apache Commons
- [x] JUnit
- [ ] Retrofit
- [ ] Android

5. Why is it important for unit tests to run quickly (select all that apply)? 1 point

- [x] to minimize the time developers wait for feedback on their code changes
- [ ] to avoid blocking the main thread in an Android app
- [ ] to minimize the startup time of the Android emulator
- [ ] to minimize the number of mistakes that developers make
- [ ] to ensure that user interface components are not slowed down

6. Which of the following is a valid, although not useful, JUnit test (select all that apply)? 1 point

- [ ]
```
@Test
public void test() {
  ensure.that(true);
}
```

- [ ]
```
@Unit
public void ensureTrue() {
  assertEquals(true,true);
}
```

- [ ]
```
@Unit
public void test() throws Exception {
  assert(true == true);
}
```

☐
```
public void test() {
  ensureTrue(true);
}
```

☑
```
@Test
public void ensureTrue() {
  assertEquals(true, true);
}
```

7. Which of the following is not an appropriate way to setup the state for MyObject in the test below (select all that apply)? 1 point

◉
```
private MyObject myObject;

@Test
public void testInitialName() {
  myObject = new MyObject();
  assertEquals("bob", myObject.getName());
}

@Test
public void testInitialSize() {
  assertEquals(10, myObject.getSize());
}
```

○
```
private MyObject myObject;

@Before
public void setUp(){
  myObject = new MyObject ();
}

@Test
public void testInitialName() {
  assertEquals("bob", myObject.getName());
}

@Test
public void testInitialSize() {
  assertEquals(10, myObject.getSize());
}
```

○
```
private MyObject myObject = new MyObject();

@Test
public void testInitialName() {
  assertEquals("bob", myObject.getName());
}

@Test
public void testInitialSize() {
  assertEquals(10, myObject.getSize());
}
```

○
```
private MyObject myObject;

@Before
public void setUp(){
  myObject = new MyObject();
}

@Test
public void testInitialName() {
  assertEquals("bob", myObject.getName());
}
```

○
```
private MyObject myObject = new MyObject();

@Test
public void testInitialName() {
  assertEquals("bob", myObject.getName());
}
```

8. Assume that the MyObject.dangerous() function throws a checked exception. Which of the following are valid ways to write a test of MyObject (select all that apply). 1 point

☐
```
@Test
public void someTest() {
  MyObject obj = new MyObject();
  assertEquals(false, obj.dangerous());
}
```

☑
```
@Test
public void someTest() throws Exception {
  MyObject obj = new MyObject();
  assertEquals(false, obj.dangerous());
}
```

☐
```
@Test
public void someTest() {
  MyObject obj = new MyObject();
  try{
    assertEquals(false, obj.dangerous());
  }catch(Exception e){ e.printStackTrace(); }
}
```

☐
```
@Unit
public void someTest() throws Exception {
  MyObject obj = new MyObject();
  assertEquals(false, obj.dangerous());
}
```

- [ ]
```
@Test
public void someTest() {
  MyObject obj = new MyObject();
  try{
    assertEquals(false, obj.dangerous());
  }catch(Exception e){}
}
```

9. What are valid uses of code coverage metrics (select all that apply)? 1 point

- [ ] techniques to help developers test faster
- [ ] to determine if an app has been tested sufficiently
- [ ] to guarantee that an app is bug free
- [ ] to evaluate the quality of a development team
- [x] to determine where more testing may be needed

10. Which of the following are true about the test shown below (select all that apply): 1 point

```
public class SomeTest {
  private MyObject myObject = new MyObject();

  @Test
  public void testInitialName() {
    assertEquals("bob",myObject.getName());
  }

  @Test
  public void testInitialSize() {
    assertEquals(10,myObject.getSize());
  }
}
```

- (•) The MyObject constructor will be called at least once for each @Test annotation in the class.
- ( ) The MyObject constructor will be called a single time before testInitialName() is executed and not called before any of the other test methods.
- ( ) The MyObject constructor will be called a single time before testInitialSize() is executed and not called before any of the other test methods.
- ( ) The MyObject constructor will not be called and testInitialName will throw a null pointer exception.
- ( ) The MyObject constructor will not be called and a @Before annotation is needed to provide a method to set the initial state of myObject.

- [x] I, **Shubham Kumar**, understand that submitting work that isn't my own may result in permanent failure of this course or deactivation of my Coursera account.

Learn more about Coursera's Honor Code

Save Submit