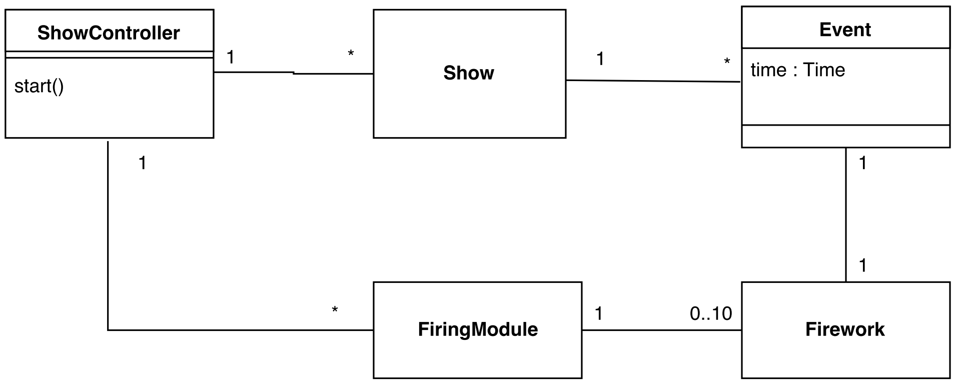
Module Title: Inf2C-SE: Introduction to Software Engineering
Exam Diet: Dec 2017
Brief notes on answers:

1. (a) *Application of knowledge.* A solution similar to the following is expected.



Marking:
2 marks: Correct notation for classes, associations, typed attribute and operation.
2 marks: Appropriate attributes and operations.
1 mark: Appropriate presentation of each class as box or attribute.
2 marks: Correct associations.
2 marks: Correct multiplicities on association ends.

(b) Answers will vary. Here are some ideas.

- *Safety:* it could be very dangerous if fireworks were accidentally set off prematurely. The system needs to be developed carefully and thoroughly tested to minimise the chance of there being bugs. Further, specific safety features could be added such as a separate mechanism for enabling or disabling all firework firing.
- *Security:* the communications between the controller and the firing modules should be secure so pranksters can't interfere and set off fireworks at potentially dangerous times.
- *Usability:* The user interface should be designed to minimise chance of errors both when programming shows and launching shows.
- *Reliability:* Control of firework launching should be dependable. For example, the system might periodically run diagnostics checking strength of wireless connections and health of firing devices.

Marking:
Up to 2 marks for each of up to 3 requirement. Only 0.5 marks for each if just general non-functional requirement kinds are named.

(c) *Bookwork.* A 3-way merge considers how files F2 and F3 are each modifications of some nearest common ancestor file F1. It creates a new file F4 that incorporates into F1 both the changes from F1 to F2 and from F1 to F3. When changes are conflicting, i.e. are to the same lines of F1, then both changes are inserted into F4 with extra text making clear the conflicting changes and prompting the developer to resolve them.

A standard usage scenario is when a checked-out version of a file, possibly locally modified, has to be brought into sync with a version of that file that has been modified in a different way, perhaps that is on a different development branch and/or that has been modified by another developer.

Marking:
3 marks for basic merge of differences from common ancestor.
1.5 marks for description of what happens when there are conflicts.
1.5 marks for a description of a plausible scenario.

(d) *Bookwork.*

(i). Maven insists on a uniform layout of files. Because of this users have to specify less configuration information and it is consequently easier for users to run various build tasks.
Marking:
1 mark for uniform file layout, 1 mark for benefit.

(ii). Examples mentioned in class are compiling unit testing, packaging, integration testing, installing, deploying, generating documentation.
Marking:
1 mark for each of 2 examples.

2. (a) *Bookwork*: Coupling is a measure of the inter-dependency between classes or components. Lower coupling is more desirable. Lower coupling means that components are more independent of each other which generally makes them easier to understand and easier to maintain. **Marking**: 2 marks for definition
1 mark for low being preferrable
2 marks for benefits full marks if both given.

(b) *Application of Knowledge, Bookwork*
The obvious invariant is:

```
//@ invariant 0 <= hour < 24 && 0 <= minute < 60;
```

Run-time checks of the invariant should be carried out at the start and end of each of these methods.
Marking:
2 marks for syntax (Special comment, `invariant` keyword, boolean-valued expression)
2 marks for sensible properties being checked
2 marks for when invariant should be checked.

(c) *Problem solving.*

```
public class SubtractTest {
    @Test
    public void test1() {
         Time t1 = new Time(3, 45);
         Assert.assertEquals(new Time(3, 44), t1.subtractMinute());
    }
    @Test
    public void test2() {
         Time t1 = new Time(3, 00);
```

```
            Assert.assertEquals(new Time(2, 59), t1.subtractMinute());
        }
    }
```

`assertEquals()` is preferred because, when the equality is false, the error message shows both the expected and actual values. If just `assertTrue` is used, JUnit does not have access to either value and therefore cannot show them if the assertion fails.

Marking:
3 marks: sensible tests
2 marks: Correct JUnit syntax (`@Test`, `Assert.assertEquals()` calls).
2 marks: Explanation of why `assertEquals` is preferred.

(d) *Bookwork.* In UP each phase is seen as involving a *combination* of activities (*workflows* or *disciplines* in UP terminology), with the blend of activities shifting as one moves through the phases. Also each UP phase is explicitly seen as involving one or more iterations, each iteration producing an increment on the previous one.

Marking:
2.5 marks for idea of phases involving blends and emphasis shifting.
1.5 marks for iteration within phases.

(e) *Bookwork.* Examples of characteristics include team being small, team having more software development experience, team willing to take initiative and respond flexibly to change rather than preferring to always follow a prescribed plan.

Marking:
1 mark each for any three reasonable points.

3. (a) *Bookwork.*

(i). A refactoring is a small change made to a program that does not change the overall function of the program.
Marking:
1 mark for small change.
1 mark for function of program not changed.

(ii). Refactorings are strongly desirable when repeated changes to code have made it hard to understand and hard to maintain. Refactorings improve understandability and maintainability.
Marking:
1 mark for a reason why code might become difficult to understand and maintain.
2 marks for issues that refactoring addresses such as understandability and maintainability.

(iii). Software development involves repeated increments being realised in the software which can quickly destroy structure and modularity if refactoring is not regularly performed.
Marking:
1 mark for relevant aspect of Agile processes
1 mark for why this causes problems that refactoring can address

(b) *Bookwork.* From lecture, the most important information is:

- Exactly what you did to cause the bug and what went wrong. Ideally you want to enough information to enable the bug reader to reproduce the bug, though this is not always possible.
- the version of the software being run and information about the environment (e.g. the OS).

Ideas on the cause are good, but these must be kept *separate* from the description of the bug. Not infrequently these ideas can be wrong, and if they are mixed in, the bug fixing process will be hampered.

Marking:
Up to 4 marks for the kinds of information.
2 marks for how ideas should be handled and why.

(c) *Problem solving.* Should `fetchAndProcessCustomerInfo()` throw an exception, there is no control over the release of the exception message text to the outside world, and the message could contain potentially sensitive information. The Monster Mitigation violated is *Establish and maintain control over all of your outputs.*

Marking:
3 marks for description of the issue
1 mark for naming the Mitigation.

(d) (i). Several sections are relevant, including:

- PUBLIC Software engineers shall act consistently with the public interest.
- CLIENT AND EMPLOYER Software engineers shall act in a manner that is in the best interests of their client and employer consistent with the public interest.
- PRODUCT Software engineers shall ensure that their products and related modications meet the highest professional standards possible.

In favour: risk of death or injury to car occupants and other on the road; risk that company will lose reputation; product has known fault.
Mitigating: Maybe risk of situation occurring is so small it will never happen in practice.

Marking:
3 marks for two sections and their brief descriptions.
3 marks for arguments in favour and mitigating.

(ii). It tells you that it is completely ethical to use the information. This information was made public by your competitor. There is no public interest argument, you are not undermining professional standards. It is a competitive market, information is valuable, companies should guard it.

Marking:
Up to 2 marks for reasonable points.