Exercise Set 7 - Liveness and Asynchrony

Exercise 1

(Each answer 0.5 points)
Answer the following questions:

a When should you consider using asynchronous invocations? **Answer:**

- *When a host object can distribute services amongst multiple helper objects.*
- *When an object does not immediately need the result of an invocation to continue doing useful work.*
- *When invocations that are logically asynchronous, regardless of whether they are coded using threads.*
- *During refactoring, when classes and methods are split in order to increase concurrency and reduce liveneproblems.*

b In what sense can a direct invocation be asynchronous? **Answer:**

If you invoke the Helper directly, but without synchronization you are able to create an asynchronous invocation with a direct invocation. In this case the Host is free to accept other requests while the Host's caller must wait for the reply.

c What is "early reply"? **Answer:**

Early reply is a feature that allows a host to perform useful activities after returning a result to the client. It means that the host can continue to work after Its answer to do some operation not useful for the client.

d What are "futures"? **Answer:**

Futures is a feature that allow a client to continue in parallel with a host until the future value is needed. It means that the client can start a process even if it doesn't need that result yet, and can go ahead until he doesn't really need that data.

e When are futures better than early replies? **Answer:**

In general the idea is that I chose futures instead early reply when I need that the Host, in the first case, or the Server, in the second case, they have to complete some extra task that are not necessarily related to whom send or receive the message. For example you could need futures when you know that a functionality take a lot of time to execute. In this case a Host can start the functionality as soon as possible and proceed with its work while the service is computing. You could use early reply when you need to modify a data structure that you always what to keep optimized. In that case an Host ask to add an element to the structure and when the operation is done the Server can reply to the invocation and then proceed with optimization of the structure.

Exercise 2

(7.5 points)