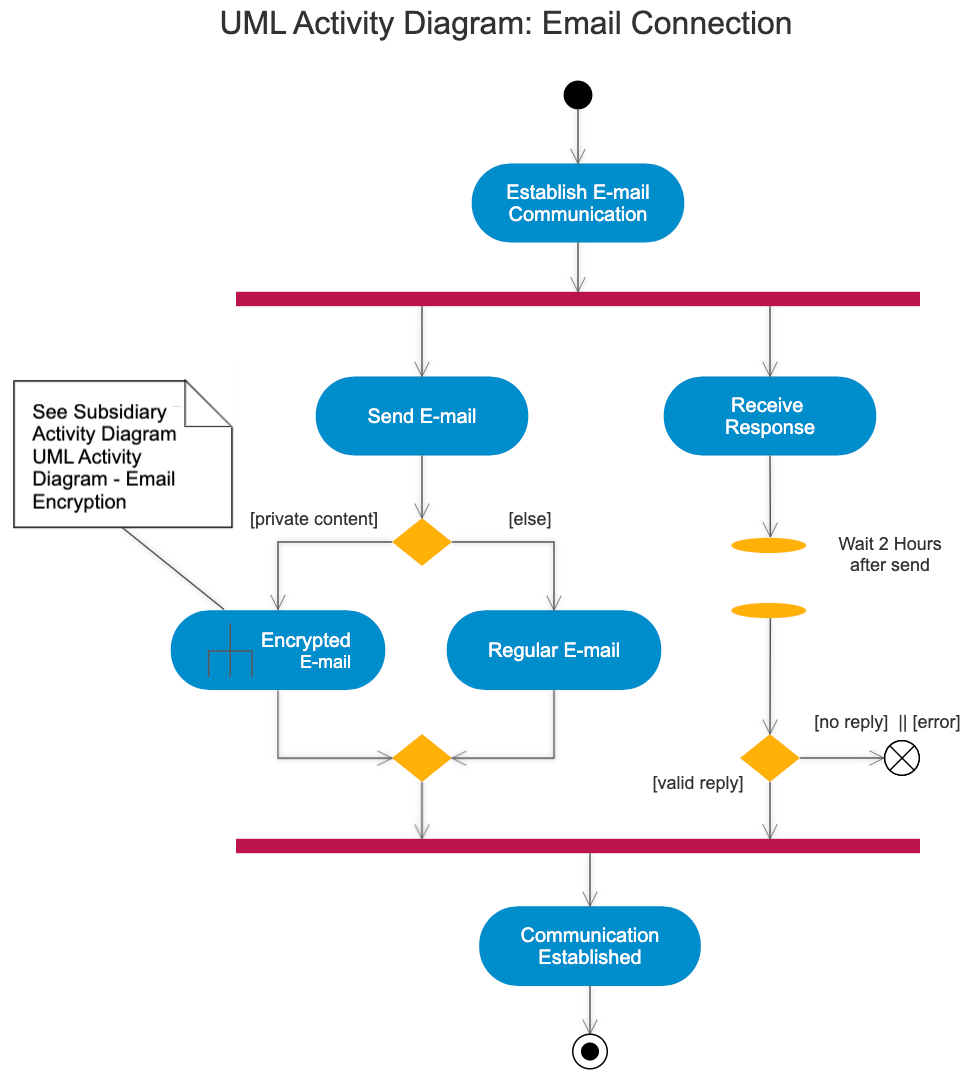
Antonio Scalfaro

CMSC 335

Professor Seely

Homework #2

1. There are 13 UML Standard types of diagrams, they are as follows:
   1. Class – Represents the structure of a system by showing attributes, classes, and relationships in the system.
   2. Activity – Represents the work flow of a system by displaying the sequence by which Objects are used in the system.
   3. Object - Represents the static structure of the system at a given time and can be used to verify a Class diagram.
   4. Use Case – Models the system functionality using actors and cases that the system could be used for
   5. Sequence – Describes interactions among classes in terms of messages that may be exchanged.
   6. Package – Organizes and structures the elements of a system to display dependencies between packages and what each package contains.
   7. State – Depicts the behavior of elements in a system, showing different states and transitions of states depending on events taking place.
   8. Component – Represents the components of a system such as libraries, executables, files, etc. and the dependencies between them.
   9. Communication – Models the interaction between objects in a system, describing both the static structure and the dynamic behavior of a system.
   10. Composite Structure – Show the internal parts of a class.
   11. Interaction Overview – A combination of activity and sequence diagrams that models the sequence of actions, allowing for decomposition of more complex interactions.
   12. Timing – A type of sequence diagram that focuses on processes that take place during a specific period of time.
   13. Deployment – Depicts the physical resources in a system including nodes, components, and connections.
   14. 
2. In this case, both toString methods would be Overridden methods and the H2ClassA and H2ClassB methods would look something like this:
   1. Public String toString() {

StringBuilder s = new StringBuilder();

For (H2ClassB b : list){

Sb.append(b).append(“ “);

Return sb.toString();}

H2ClassB toString:

Public String toString(){

Return Integer.toString(x); }

1. This code snippet can be corrected in many ways but there are two that come to mind right away. The first way could be to add a class field to H2ClassC of int a and assign its value in the constructor when passed int a. The H2ClassD would then have a field int b and a constructor that takes a parameter int a, int b and uses the super key word passing in int a then setting int b to the class field. The next case could be that H2ClassC has a field int a that is set and the constructor takes no parameters. Meanwhile, the H2ClassD class would have a field int b and the constructor would accept a parameter and set it to int b after it calls super.
2. This code snippet throws a compilation error because it tries to call another constructor using this() but it is not the first line in the constructor. Swapping lines 5 and 6 would fix this issue by calling this(5, 12) before any other code in the constructor.
3. The assignment of a double value into an int declaration is invalid. It can be fixed in a few ways, either you can change the type declaration to a double or switch the value assigned to an int value (17).
4. The code snippet is incorrect because the variable x is declared to be final, which means that it must be assigned a value upon instantiation or in every constructor in the class. Since the first constructor does not set a value for x, it will cause a compilation error. Either change the variable x to not be final or include an assigned value to x in all constructors.
5. Two issues are prevalent with this code snippet, the constructor has a return value and the final int variable, x, is not assigned a value. The constructor should be changed to have no return value and x should be assigned a value either in the constructor or when it is instantiated.
6. This code snippet is incorrect because the H2Classl does not have a constructor to create an instance of the class and it has no way to set a value for the final int x. Adding a constructor to the class and setting the value of x by passing a parameter to the constructor is one way to give the value 24 to x. Another way of assigning the value to x could be done by giving the field the assigned value of 24, making every instance of H2Classl having the value of 24 for x.
7. This snippet will not work because MouseListener is an interface with five methods and the addMouseListener method expects all five of those methods to be implemented, this snippet implements only one. You can fix this by implementing all five of the methods or by changing the call of MouseListener to a call to MouseAdapter, which allows for the implementation of only methods you need.
8. The FX GUI code has a few issues wrong with it. The first is the import is incorrect, javax.javafx.\* should be changed to javafx.\*. Next, the event handler is not contained inside of a method. This event handler could be enclosed in a method that gets called upon the event or it could reside inside the start method but it must be contained in a method.