Coding – Things to Learn

Here is a short list of things you should know (or at least be quite familiar with):

# Design patterns

* The ‘gang of four’ wrote about 23 design patterns found in good code.
* Russ Olsen has a book (Design patterns in ruby) explaining 14 of these in a ruby context

# Design principles (SOLID, DRY, SRP, DIP)

## DRY

* **D**on’t **R**epeat **Y**ourself – if you find yourself doing the exact same thing 2 or more times, you can probably find a more efficient way to do it.

## SRP

* **S**ingle **R**esponsibility **P**rinciple – every class, module, object or method should be responsible for exactly *one* part of the functionality of the program

## DIP

* **D**ependency **I**nversion **P**rinciple – high level modules should not depend on low-level modules. Both should depend on abstractions

# Methods of working (Agile, Scrum, Kanban, Waterfall)

## Waterfall

## Agile

## Scrum

## Kanban

# Disciplines (TDD, BDD, OOP, CI, Pair Programming)

## Test-Driven Development

## Behaviour Driven Development

## Object-oriented programming

## Pair Programming

## Continuous Integration

* Integrate code into a shared repository several times each day→allows people to detect and locate errors early and quickly

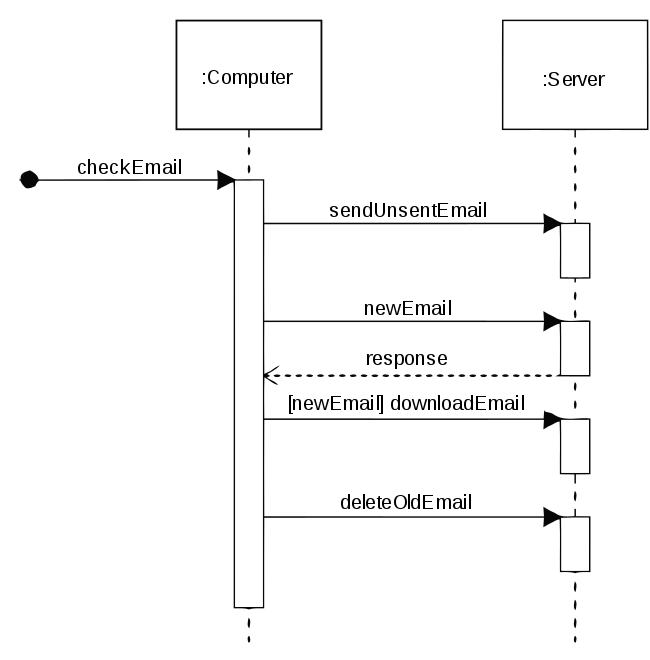
# Artifacts (UML, sequence diagrams, flow charts, decision tables)

* An **artefact** is a tangible by-product produces during the process of creating software: e.g. data models, diagrams, design documents, any *thing* that gets created (usually not counting code)
  + There’s usually a list of specific required artifacts that must be produced

## UML

* A standard language for specifying, visualising, constructing and documenting the artifacts of software systems.
* The below are often created in UML (I think)

## Sequence Diagrams

* Shows objects and their interactions (e.g. messages passed) arranged in a time sequence
  + Simultaneous processes are represented as parallel vertical lines
  + Messages exchanged represented by horizontal arrows
* 

## Flow Charts

* Self-explanatory – presumably flow charts of the process

## Decision Tables

* Used in software *testing*, meant to test different input combinations→represent the corresponding system behaviour
  + Also known as cause-effect table