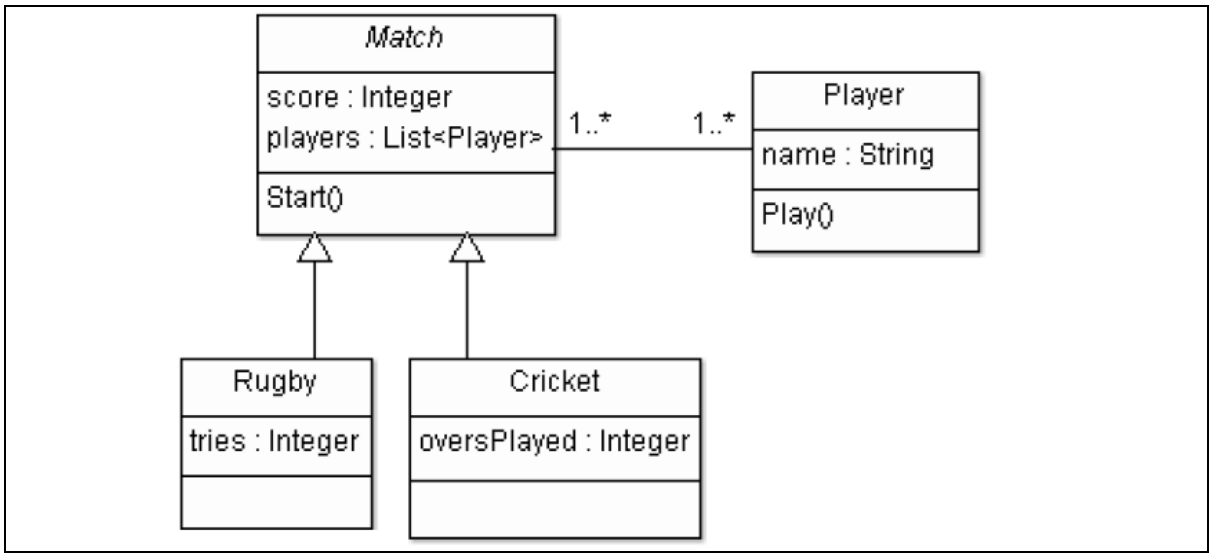
# Tutorial 3 – Inheritance

This week we will look at classes, interfaces and abstraction, with a particular focus on inheritance. We'll use UML diagrams and standards to understand and answer questions about a system.

1: Examine the following simple relationship in the following diagram:




a. What type of diagram is this?

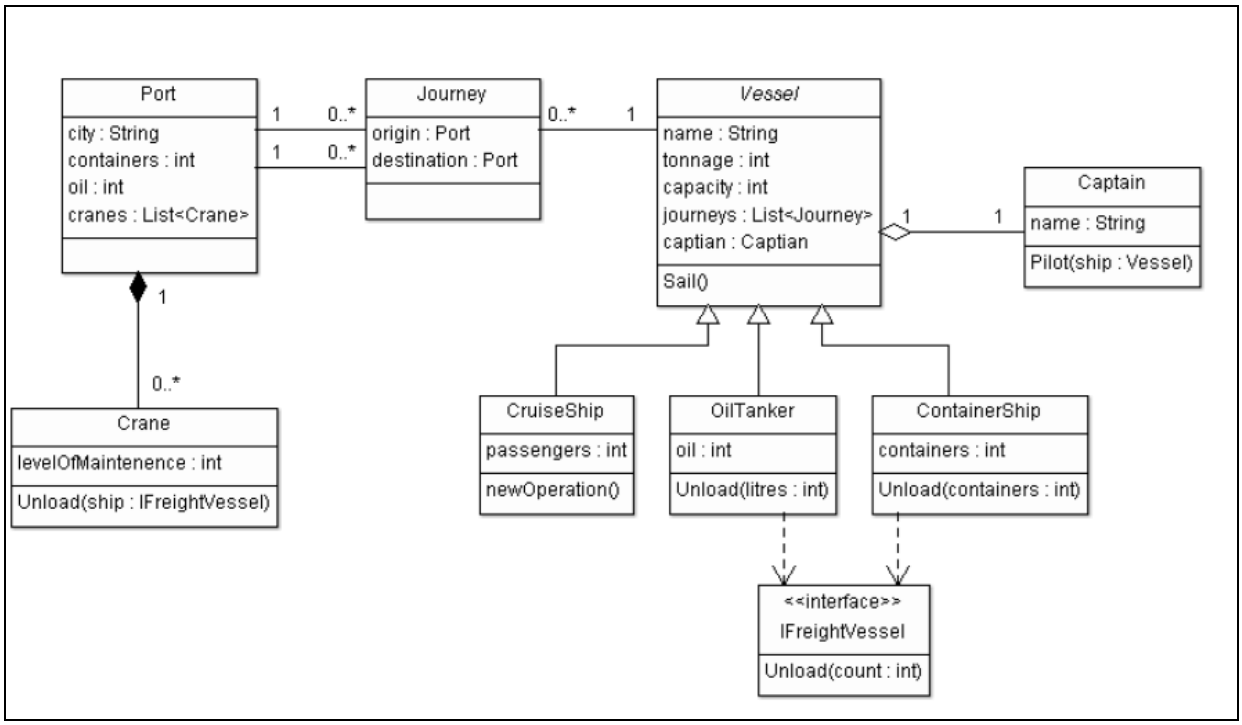
b. What is the relationship between the Rugby and Match Classes?

c. What is the multiplicity between Match and Player?

d. What type of class do you think Match is likely to be?

# Expressive Relationships in UML

The following UML class diagram describes a way to model sea ports and large ships. We'll be examining this example in detail as it includes the association, aggregation, composition and interface relationships.




2. Name two classes that have an Association relationship

3. Name two classes that have an Aggregation relationship

4: Name two classes that have a Composition relationship

Refer to the class diagram on page 2 for the following questions.

5. Name a class that Implements an interface

6. Name a class that extends another class (i.e. name a subclass)

7. Identify one abstract class

8. List all Attributes of the OilTanker class (including inherited ones)

9. Reference types are the different variable types that can be used to refer to an object and are related to polymorphism. What reference types can be used to refer to a ContainerShip object?

# Creating an inheritance hierarchy

Given is the abstract class Vehicle and the classes Car and Bicycle. Think carefully about what the relationship between the classes are.

Every vehicle has a name of type String, a production year of type int and can change gear, accelerate and brake (The car and bicycle classes implement this in their own way).

However, only Car objects have unique attributes of engine size of type String and can turn on their headlights and turn on their windscreen wipers which return Boolean values.

Similarly, bicycles have a unique attribute cadence (no. of revolutions per minute) of type int, as well as a number of unique responsibilities which including ringing a bell and changing the cadence which also return Boolean values.

Complete the skeleton code by writing the instance variable and method declarations as well as the constructor for each class. (NB: You do not actually have to implement each method, simply give the declaration).

Vehicle Class:

```
public abstract class Vehicle {

//TODO

Public Vehicle(//TODO) {

//TODO
}

//Method declarations TODO
```

Car Class:

```
Public class Car //TODO


//TODO
```

Bicycle Class:

```
Public class Bicycle //TODO


//TODO
```