Object-Oriented Programming

Features of OOP Languages

- Encapsulation of Data and Code – Data and procedures acting on that data (methods) grouped together within types (classes). Instances of these classes are objects.
- Inheritance – Can define new, 'child', classes based on existing, 'parent', classes. Child class inherits data fields and methods of parent
- Subtyping with Dynamic Dispatch – Objects of child class can appear wherever parent is expected. Dynamic dispatch means that the choice of which method to call – the parent's or child's – is made at runtime.

Subset interpretation of Subtyping

- A type defines a set of values. A subtype defines a set of values that is a subset of the set defined by its parent type.
- A subclass inherits all methods and fields of superclass.
- If the values of S are a subset of T, then an expression expecting T values will not be unpleasantly surprised to receive only S values.
- i.e. If S <: T , then every value of type S is also a value of T.
- In java:
  - As we go up the inheritance chain a class has fewer and fewer methods and fields (width subtyping), until we reach Object, the supertype of all classes, which has the fewest. Thus for all class types C in Java, C <: Object. The interpretation of subtyping as subsets holds: every object that has a type lower in an inheritance hierarchy also has a type higher in the hierarchy, but not vice versa.

Subtyping on functions

- Covariant subtyping: If B <: A, then C -> B is a subtype of C -> A.
  - The relative order of subtyping of functions is the same as relative order of subtyping of classes
- Contravariant subtyping: If B <: A, then A -> C is a subtype of B -> C.

Subtyping in functions

- $S_1 \rightarrow S_2 <: T_1 \rightarrow T_2$ if $T_1 <: S_1$ and $S_2 <: T_2$
- $S_1 \rightarrow S_2$ means function is expecting an argument of type S1 and returns a value S2.
- -> type constructor is covariant in return type and contravariant in argument type