

What is the problem that generics solves?

Brief note on generics and generic types

Dr Tom Ridge

2018 Q1

Look at this code:

```
1 List l1 = cons(1,cons(2,cons(3,nil())));
2 List l2 = cons(true,cons(true,cons(false,nil())));
```

- ▶ Notice that the lists l1 and l2 are different: l1 contains integers, and l2 contains booleans.
- ▶ What is the type of hd(l1)? What is the type of hd(l2)?
- ▶ All Java knows is that these are lists, so all Java knows about the elements are that they are objects
- ▶ Java cannot “see” that the head of l1 must be an integer, and the head of l2 must be a boolean.

Some syntax: List<Integer>

- ▶ Till now we have been using plain Java List.
- ▶ This is fine; but all it tells you about the elements of the list is that they are Objects.
- ▶ Early Java supported only this syntax. But it was known for a long time (30 years or more) that this was not sufficient.
- ▶ What is needed is for the “list” type to somehow tell you what the type of the elements is.
- ▶ For this reason, the syntax List<Integer> (or more generally List<X> for some type X) was introduced. This type means “list of integers”.
- ▶ The type List<X> is a “generic type” which can be “instantiated” to give a particular type such as List<Integer>.
- ▶ When you take the head of a List<Integer>, Java knows that that the object is an Integer. You no longer have to cast it to a particular desired type as with Integer i = (Integer)(hd(xs)).

Example

```
1 List<Integer> xs = cons(1,cons(2,cons(3,nil())));
2 int x = hd(xs); // no casting needed!
3 // etc
```

Resources

Make sure you read and understand the following tutorials:

- ▶ Oracle tutorial
<https://docs.oracle.com/javase/tutorial/java/generics/index.html>
- ▶ Extra Oracle tutorial
<https://docs.oracle.com/javase/tutorial/extra/generics/index.html>