# Generics

* A type parameter can be added to any class or method.
* The type parameter makes it possible to use the class or method with several different kind of objects (not built-in types, however).
* The type parameter is expressed inside angle brackets, i.e. <E> just after the return type. *public <E> void aMethod (…)*. There can be multiple type parameters in which case they are separated by commas.
* When using a method with type parameters the name of the type parameter is replaced with the real type.
* Type parameters are heavily used in data structures and functional interfaces.
* I use type parameters in every single data structure I utilize. Before type parameters I had to utilize polymorphism and put references of class Object into the data structures. This led to a huge amount of type casts.
* Cannot be used with arrays. See why: *https://www.tothenew.com/blog/why-is-generic-array-creation-not-allowed-in-java/*.
* Type parameters do not inherit! DS<Object> is not the superclass of DS<String>.
* You can however tell the compiler that you will use super classes or sub classes of the actual parameter:
  + <? extends SomeType> means you can use the type SomeType and classes extending or implementing SomeType. This is a so called *upper bound wildcard*.
  + <? super SomeType> means you can use type SomeType and all super classes of SomeType. This is a so called *lower bound wildcard*.
* Generics can also be applied to classes in which case the type parameter can be applied to properties and methods.
* For those interested:
  + http://www.angelikalanger.com/
  + <https://stackoverflow.com/questions/2723397/what-is-pecs-producer-extends-consumer-super>
* **One really important thing to remember is that the real type, you use in place of the generic one, must support the operation!!!**
* Examples: Usages, Generics.