



J12 Generics

Generics

Generics

Sometimes it is useful to parameterize a class with a type, T.

Rather than `IntContainer`, `LongContainer`, etc. we can just write `Container<T>`, and then create instances of types such as `Container<Integer>`.

We can also create generic methods that accept type parameters:

```
static <T> void acceptSomeValue(T value) { ... }
```

Prior to the introduction of Java generics, programmers often used `Object` as a work-around as it can refer to any non-primitive type.

Type Parameters

- By default, the only thing that is assumed about a type parameter `T` is that it is an object: i.e. it extends `Object`.
 - No primitives can be used as a generic type (big part of the reason for boxing primitives)
 - When working with a variable that has a generic type, all we can do is pass it around and call methods that are defined for `Object`.
- *Bounds* can be put on type parameters to make them “less generic”.
 - E.g., `public <T extends Number & MyInterface> void method(T t) {...}`
 - This **restricts** the types that can be used with the generic.
 - This **increases** the assumptions that can be made about a variable of this generic type.