Opening Discussion

- ACM Meeting
- Minute essay comments:
  - Falling behind?
  - What is feasible in projects?
- IcP code.
Scala allows you to pass in type parameters similar to normal arguments.

- Type parameters go in [] instead of ()
- We have seen these on types like List and Array already.
Type parameters provide us with parametric polymorphism, another form of Universal polymorphism.

Code will work on any appropriate type arguments.
You can add type parameters after the name of a class or a trait.

This specifies an unknown type that instances should work with.

The actual type is specified at creation. Not always needed if inference system can figure out from arguments.

This is how you make collections that can hold any type.
You can also add type parameters after function/method names.

These specify types that the function/method will work on.

These are rarely specified as they should be identifiable from the arguments.
A parametric type of \([A]\) can only be used in a way that is applicable to Any because \(A\) could be anything.

You can place bounds on the type with <: and >:. The <: is far more common. \([A <: B]\) means that \(A\) must be a subtype of \(B\) to be used.

The <\% indicates you accept things with implicit conversions.
Ordered Trait

- This trait gives you all comparison operators if you just implement a compare method.
- It is parametric on the type to compare to (typically the inheriting type).
- This trait is good for anything that has a natural order to it.
Let's take what time we have left and continue putting pieces into the drawing program.

Hopefully we can get the command processor working.
Next class we will look at some other collection types and explore the polymorphism of Scala collections. What aspects of polymorphism (inclusion or parametric) do you have questions about?