- Is it obvious that an A *is-a* B?
- Will clients want to use an A as a B?

Dynamically, meaning method implementation is determined at runtime based on the class of the object, not the type of the variable or expression.

They compose operators and combine elements of some domain into new elements.

It is required for all members (*i.e.*, it must also be applied to fields).

It is optional if you're implementing an abstract member.

They invoke the primary constructor.

Each non-auxiliary constructor must call another constructor as its first line?

Abstract members.

As in Java, with `final` keyword on class or method.

You can also use `sealed`, which only allows derivation in the same source file.

Composition is preferred if only code re-use is desired.

When the function is pure, i.e. has no side effects and does not depend upon mutable state.

A field defined as a class parameter allows visibility modifiers, `val` or `var`, and `override`.

```
class ArrayElement(
  val contents : Array[String]
) extends element
```

Private members and overriden members.

```
def width() : Int
def width : Int
```

If defined parameterless, it must be called that way.

This is to support the uniform access principle.

It causes inadvertent breakage of subclasses by changing a superclass.

Forcing override notation helps solve it.

- Java: fields, methods, types, packages
- Scala: values (fields, methods, packages, singletons), types (class and trait names)

It allows the overriding of parameterless and empty-paren methods with a val.

Use companion object to provide methods for making instances of the companion class.

Then client code can import just the factory methods (which share the same name).

Optionally, make concrete subclasses private inside singleton to hide details. Only the visible superclass is returned.

To allow importing packages themselves and not just the names of types.

This also allows the importing of fields and methods of singleton objects.

It allows empty-paren methods to override parameterless ones and vice-versa.