```
import ObjectiveC.List
import ObjectiveC.NSObjCRuntime
import ObjectiveC.NSObject
import ObjectiveC.Object
import ObjectiveC.Protocol
import ObjectiveC.hashtable
import ObjectiveC.hashtable2
import ObjectiveC.message
import ObjectiveC.objc
import ObjectiveC.objc_api
import ObjectiveC.objc_auto
import ObjectiveC.objc_class
import ObjectiveC.objc_exception
import ObjectiveC.objc_load
import ObjectiveC.objc_runtime
import ObjectiveC.objc_sync
import ObjectiveC.runtime
import _Concurrency
import _StringProcessing
import _SwiftConcurrencyShims

@available(macOS 10.0, iOS 1.0, tvOS 1.0,
watchOS 1.0, *)
@frozen public struct NSZone {
}

@available(macOS 10.0, iOS 1.0, tvOS 1.0,
watchOS 1.0, *)
extension NSZone : BitwiseCopyable {
}

/// The Objective-C BOOL type.
///
```

```
/// On 64-bit iOS, the Objective-C BOOL
type is a typedef of C/C++
/// bool. Elsewhere, it is "signed char".
The Clang importer imports it as
/// ObjCBool.
@available(macOS 10.0, iOS 1.0, tvOS 1.0,
watchOS 1.0, *)
@frozen public struct ObjCBool :
ExpressibleByBooleanLiteral, Sendable {

    public init(_ value: Bool)

    /// The value of `self`, expressed as
a `Bool`.
    public var boolValue: Bool { get }

    /// Create an instance initialized to
`value`.
    public init(booleanLiteral value:
Bool)

    /// A type that represents a Boolean
literal, such as `Bool`.
    @available(iOS 1.0, tvOS 1.0, watchOS
1.0, macOS 10.0, *)
    public typealias BooleanLiteralType =
Bool
}

@available(macOS 10.0, iOS 1.0, tvOS 1.0,
watchOS 1.0, *)
extension ObjCBool : CustomReflectable {
```

```
    /// Returns a mirror that reflects
`self`.
    public var customMirror: Mirror { get
}
}

@available(macOS 10.0, iOS 1.0, tvOS 1.0,
watchOS 1.0, *)
extension ObjCBool :
CustomStringConvertible {

    /// A textual representation of
`self`.
    public var description: String {
get }
}

@available(macOS 10.0, iOS 1.0, tvOS 1.0,
watchOS 1.0, *)
extension ObjCBool : BitwiseCopyable {
}

/// A Sequence of AnyClass
@available(macOS 13.0, iOS 16.0, tvOS
16.0, watchOS 9.0, *)
public struct ObjCClassList : Sequence {

    /// A type representing the
sequence's elements.
    public typealias Element = AnyClass

    /// A type that provides the
sequence's iteration interface and
```

```
    /// encapsulates its iteration state.
    public class Iterator :
IteratorProtocol {

        /// Advances to the next element
and returns it, or `nil` if no next
element
        /// exists.
        ///
        /// Repeatedly calling this
method returns, in order, all the
elements of the
        /// underlying sequence. As soon
as the sequence has run out of elements,
all
        /// subsequent calls return
`nil`.
        ///
        /// You must not call this method
if any other copy of this iterator has
been
        /// advanced with a call to its
`next()` method.
        ///
        /// The following example shows
how an iterator can be used explicitly to
        /// emulate a `for`-`in` loop.
First, retrieve a sequence's iterator,
and
        /// then call the iterator's
`next()` method until it returns `nil`.
        ///
        ///     let numbers = [2, 3, 5,
```

```
7]
        ///     var numbersIterator =
numbers.makeIterator()
        ///
        ///     while let num =
numbersIterator.next() {
        ///         print(num)
        ///     }
        ///     // Prints "2"
        ///     // Prints "3"
        ///     // Prints "5"
        ///     // Prints "7"
        ///
        /// - Returns: The next element
in the underlying sequence, if a next
element
        ///   exists; otherwise, `nil`.
        public func next() -> AnyClass?

        /// The type of element traversed
by the iterator.
        @available(iOS 16.0, tvOS 16.0,
watchOS 9.0, macOS 13.0, *)
        public typealias Element =
AnyClass
    }

    /// Returns an iterator over the
elements of this sequence.
    public func makeIterator() ->
ObjCClassList.Iterator
}
```

```
/// Tells `objc_enumerateClasses` which
images to search.
@available(macOS 13.0, iOS 16.0, tvOS
16.0, watchOS 9.0, *)
public enum ObjCEnumerationImage {

    case dynamicClasses

    /// Search dynamically registered
classes
    case image(UnsafeRawPointer)

    /// Search the specified image (given
a handle from dlopen(3))
    case machHeader(UnsafeRawPointer)
}

/// The Objective-C SEL type.
///
/// The Objective-C SEL type is typically
an opaque pointer. Swift
/// treats it as a distinct struct type,
with operations to
/// convert between C strings and
selectors.
///
/// The compiler has special knowledge of
this type.
@available(macOS 10.0, iOS 1.0, tvOS 1.0,
watchOS 1.0, *)
@frozen public struct Selector :
ExpressibleByStringLiteral, @unchecked
Sendable {
```

```
    /// Create a selector from a string.
    public init(_ str: String)

    /// Create an instance initialized to
`value`.
    public init(stringLiteral value:
String)

    /// A type that represents an
extended grapheme cluster literal.
    ///
    /// Valid types for
`ExtendedGraphemeClusterLiteralType` are
`Character`,
    /// `String`, and `StaticString`.
    @available(iOS 1.0, tvOS 1.0, watchOS
1.0, macOS 10.0, *)
    public typealias
ExtendedGraphemeClusterLiteralType =
String

    /// A type that represents a string
literal.
    ///
    /// Valid types for
`StringLiteralType` are `String` and
`StaticString`.
    @available(iOS 1.0, tvOS 1.0, watchOS
1.0, macOS 10.0, *)
    public typealias StringLiteralType =
String
```

```
    /// A type that represents a Unicode
scalar literal.
    ///
    /// Valid types for
`UnicodeScalarLiteralType` are
`Unicode.Scalar`,
    /// `Character`, `String`, and
`StaticString`.
    @available(iOS 1.0, tvOS 1.0, watchOS
1.0, macOS 10.0, *)
    public typealias
UnicodeScalarLiteralType = String
}

@available(macOS 10.0, iOS 1.0, tvOS 1.0,
watchOS 1.0, *)
extension Selector : Equatable, Hashable
{

    /// Returns a Boolean value
indicating whether two values are equal.
    ///
    /// Equality is the inverse of
inequality. For any values `a` and `b`,
    /// `a == b` implies that `a != b` is
`false`.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
compare.
    public static func == (a: Selector,
b: Selector) -> Bool
```

```
    /// Hashes the essential components
of this value by feeding them into the
    /// given hasher.
    ///
    /// Implement this method to conform
to the `Hashable` protocol. The
    /// components used for hashing must
be the same as the components compared
    /// in your type's `==` operator
implementation. Call `hasher.combine(_:)`
    /// with each of these components.
    ///
    /// - Important: In your
implementation of `hash(into:)`,
    ///   don't call `finalize()` on the
`hasher` instance provided,
    ///   or replace it with a different
instance.
    ///   Doing so may become a compile-
time error in the future.
    ///
    /// - Parameter hasher: The hasher to
use when combining the components
    ///   of this instance.
    public func hash(into hasher: inout
Hasher)

    /// The hash value.
    ///
    /// Hash values are not guaranteed to
be equal across different executions of
    /// your program. Do not save hash
```

```
values to use during a future execution.
    ///
    /// - Important: `hashValue` is
deprecated as a `Hashable` requirement.
To
    ///   conform to `Hashable`,
implement the `hash(into:)` requirement
instead.
    ///   The compiler provides an
implementation for `hashValue` for you.
    public var hashValue: Int { get }
}

@available(macOS 10.0, iOS 1.0, tvOS 1.0,
watchOS 1.0, *)
extension Selector :
CustomStringConvertible {

    /// A textual representation of
`self`.
    public var description: String {
get }
}

@available(macOS 10.0, iOS 1.0, tvOS 1.0,
watchOS 1.0, *)
extension Selector : CustomReflectable {

    /// Returns a mirror that reflects
`self`.
    public var customMirror: Mirror { get
}
}
```

```
@available(macOS 10.0, iOS 1.0, tvOS 1.0,
watchOS 1.0, *)
extension Selector : BitwiseCopyable {
}

@available(macOS 10.0, iOS 1.0, tvOS 1.0,
watchOS 1.0, *)
@inlinable public func
autoreleasepool<Result>(invoking body: ()
throws -> Result) rethrows -> Result

/**
 * Enumerates classes, filtering by
image, name, protocol conformance and
superclass.
 *
 * - Parameter fromImage: The image to
search; defaults to the caller's image.
 * - Parameter matchingNamePrefix: If
specified, a required prefix for the
class name.
 * - Parameter conformingTo: If
specified, a protocol to which the
enumerated classes must conform.
 * - Parameter subclassing: If specified,
a class which the enumerated classes must
subclass.
 *
 * - Returns: A `Sequence` of classes
that match the search criteria.
 */
@available(macOS 13.0, iOS 16.0, tvOS
```

```
16.0, watchOS 9.0, *)
public func
objc_enumerateClasses(fromImage:
ObjCEnumerationImage
= .machHeader(#dsohandle),
matchingNamePrefix: String? = nil,
conformingTo: Protocol? = nil,
subclassing: AnyClass? = nil) ->
ObjCClassList

@available(macOS 10.0, iOS 1.0, tvOS 1.0,
watchOS 1.0, *)
extension NSObject : Equatable, Hashable
{

    /// Returns a Boolean value
indicating whether two values are
    /// equal. `NSObject` implements this
by calling `lhs.isEqual(rhs)`.
    ///
    /// Subclasses of `NSObject` can
customize Equatable conformance by
overriding
    /// `isEqual(_:)`. If two objects are
equal, they must have the same hash
    /// value, so if you override
`isEqual(_:)`, make sure you also
override the
    /// `hash` property.
    ///
    /// - Parameters:
    ///   - lhs: A value to compare.
    ///   - rhs: Another value to
```

```
compare.
    public static func == (lhs: NSObject,
rhs: NSObject) -> Bool

    /// The hash value.
    ///
    /// `NSObject` implements this by
returning `self.hash`.
    ///
    /// `NSObject.hashValue` is not
overridable; subclasses can customize
hashing
    /// by overriding the `hash`
property.
    ///
    /// **Axiom:** `x == y` implies
`x.hashValue == y.hashValue`
    ///
    /// - Note: the hash value is not
guaranteed to be stable across
    ///   different invocations of the
same program.  Do not persist the
    ///   hash value across program runs.
    @nonobjc public var hashValue: Int {
get }

    /// Hashes the essential components
of this value by feeding them into the
    /// given hasher.
    ///
    /// NSObject implements this by
feeding `self.hash` to the hasher.
    ///
```

```
    /// `NSObject.hash(into:)` is not
overridable; subclasses can customize
    /// hashing by overriding the `hash`
property.
    public func hash(into hasher: inout
Hasher)
}

@available(macOS 10.0, iOS 1.0, tvOS 1.0,
watchOS 1.0, *)
extension NSObject : CVarArg {
}
```