# Java Basics

Java Tutorials: Learning the Java Language

# Basic usage

- Hello World
  - Build basics
  - File naming conventions
- variables/operators
  - primitives
  - arrays: type[] var = new type[]
  - potentially new: pre/postfix, ternary, instanceof
- control flow
- syntax/semantics
- control flow

# Objects

- Definition
- Instantiation
  - Arrays: does not call default constructor!
- methods/fields
  - public, private
  - variable categories
- this
- Overloading

# Inheritance: basics

- Constructors
  - Default
  - Super
  - Implicit call to no-arg constructor
- Overriding
  - final
- Calling parent methods

# Interfaces and abstract classes

- Interfaces
  - Defined using “interface”
  - Define static fields
  - Method definitions
- Abstract classes
  - defined using “abstract”
  - Normal class
  - Inclusive of methods definitions marked as abstract
  - If subclass doesn’t override all abstract methods, then subclass must also be declared abstract

# Inheritance: extensions

- extends
- implements
- abstract

# Polymorphism

- Can operate with the most base of all classes
  - Bike bike = new MountainBike()
  - Can use type declarations that are interfaces or abstract!

# Libraries

Java 8 API

# Numbers

- Java defines classes to use numbers
- Object support for otherwise primitive types
- Most useful for:
  - maintaining object APIs
  - type conversion

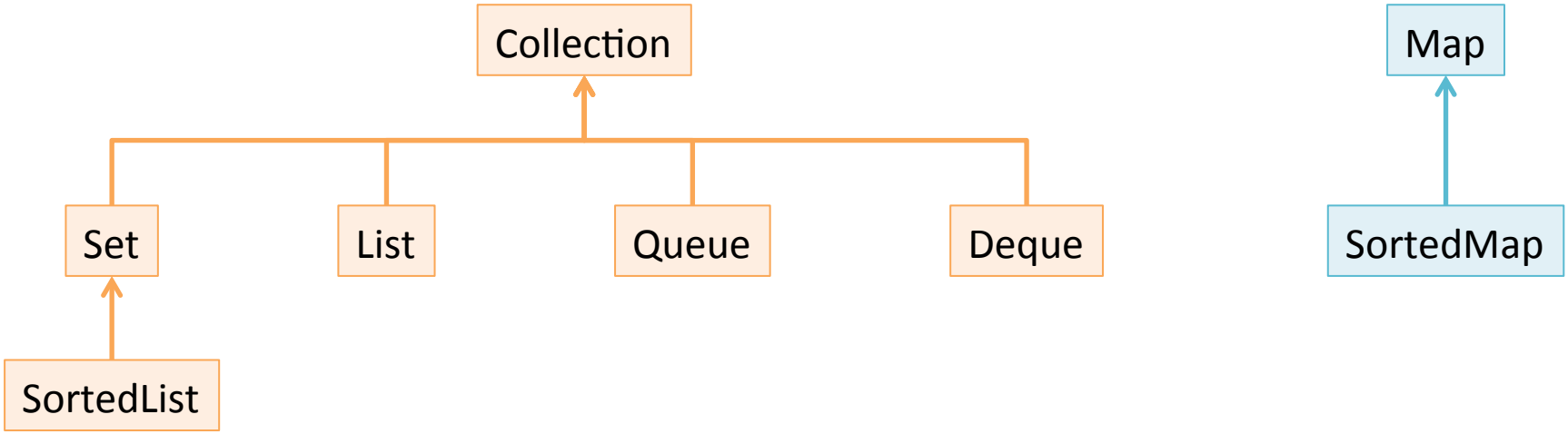
| Method | Semantics |
|---|---|
| static int parseInt(String str) | Primitive integer from a string |
| String toString() | String version of object |
| static String toString(int) | |
| static Integer valueOf(int i) | Returns an Integer object holding the value of the specified primitive |
| static Integer valueOf(String s) | |

# Strings

- Support for sequences of characters
- Don’t need explicit new
  - Why sometimes clubbed with primitives
- Immutable!
- StringBuilder
  - Mutable strings
  - Useful (and only useful) for lots of string appends

# Collections

- Set of interfaces that define various “collection” types
  - Arrays with different semantics
- Java provides
  - Interface definitions
  - Implementations of each



# Collections: Iterating

**for-each**

- Syntactically beautiful

**iterators**

- Another interface!
  - boolean hasNext
  - E next
  - void remove
- Use
  - need to remove during iteration
  - iterate in parallel

# Collections: Sorting

- Implement the Comparable interface!
- returns
  - negative integer: less-than
  - 0: equal-to
  - positive integer: greater-than
- Sorted collections require comparable objects

```
public interface Comparable<T> {
    public int compareTo(T o);
}
```

# Collections: algorithms

**Why**

- Apply common algorithms over collections
- No need to implement your own
- Requires understanding of interfaces!

**Actions**

| | |
|---|---|
| sort | comparable |
| shuffle | random |
| manipulate | |
| search | comparator |
| compose | |
| extreme values | |