Overview of Java's Support for Data & Control Abstraction

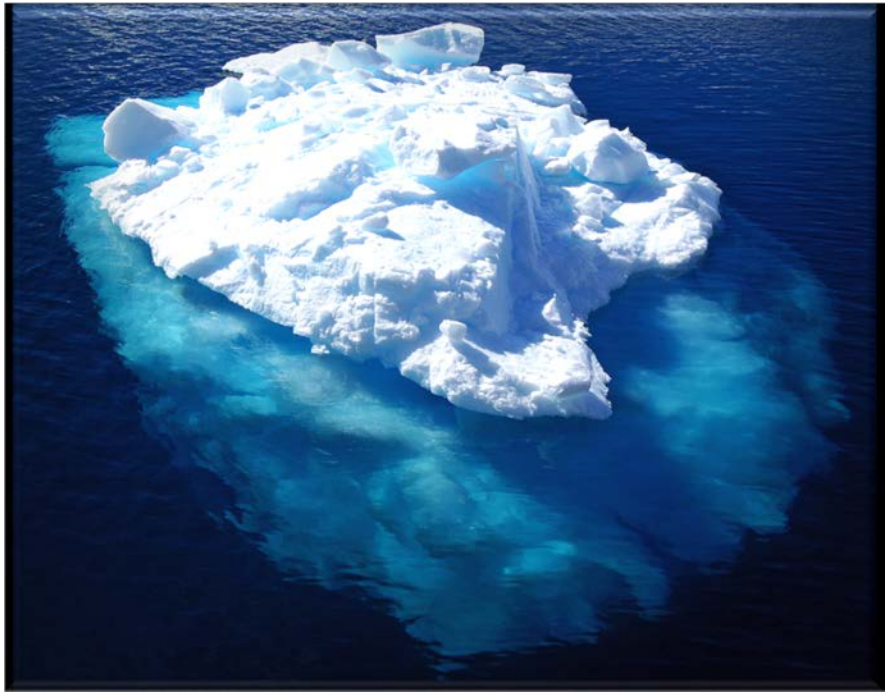
Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science

Institute for Software
Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Lesson

- Understand what the object-oriented (OO) concept of abstraction means

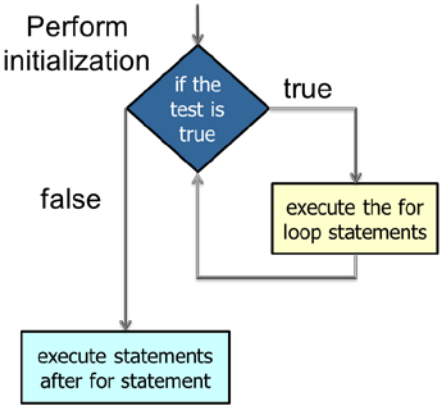


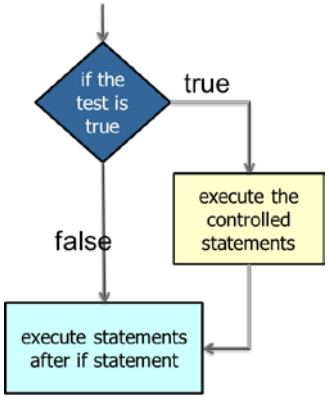
Learning Objectives in this Lesson

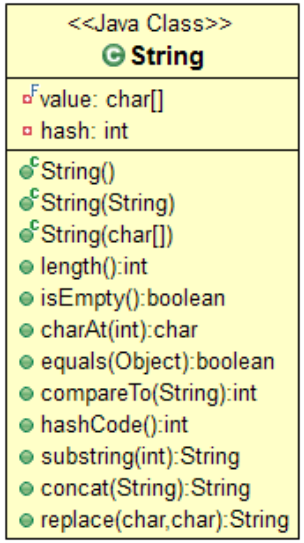
- Understand what the object-oriented (OO) concept of abstraction means
- Know the benefits abstraction provides developers of Java apps in Android

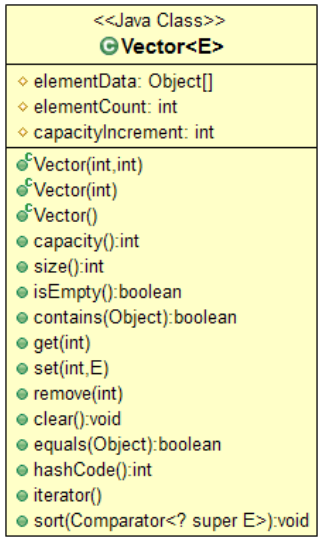
Learning Objectives in this Lesson

- Understand what the object-oriented (OO) concept of abstraction means
- Know the benefits abstraction provides developers of Java apps in Android
- Identify Java features that implement Java's data & control abstractions



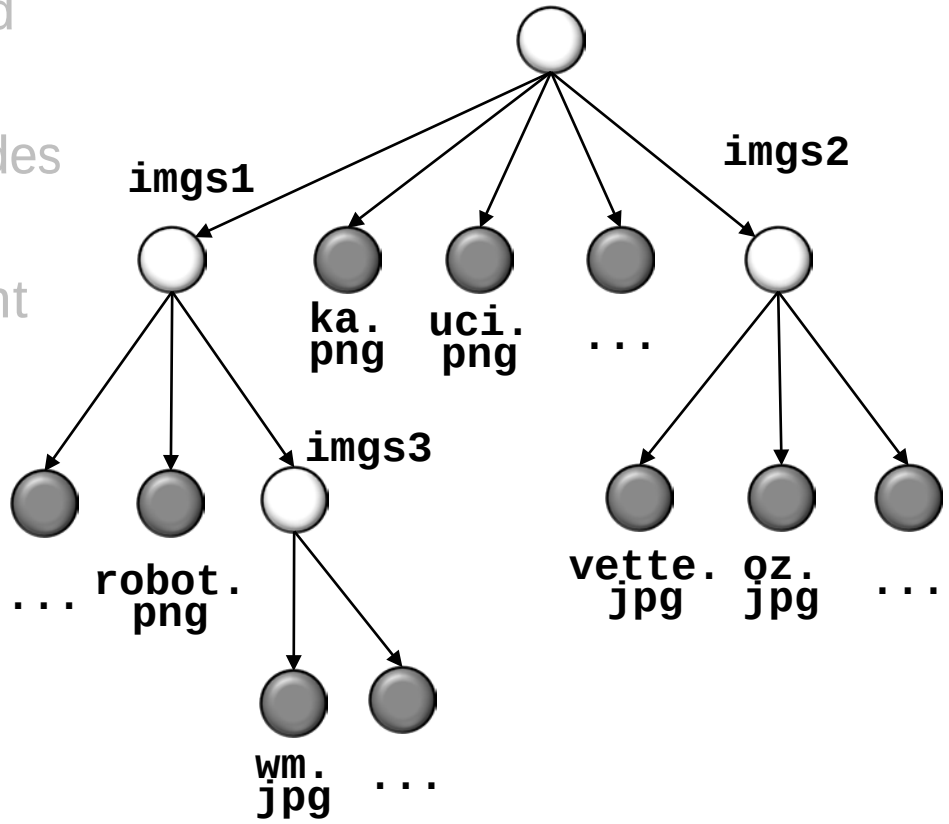






Learning Objectives in this Lesson

- Understand what the object-oriented (OO) concept of abstraction means
- Know the benefits abstraction provides developers of Java apps in Android
- Identify Java features that implement Java's data & control abstractions
- Recognize how to apply these abstractions to count the # of images accessed via a web page

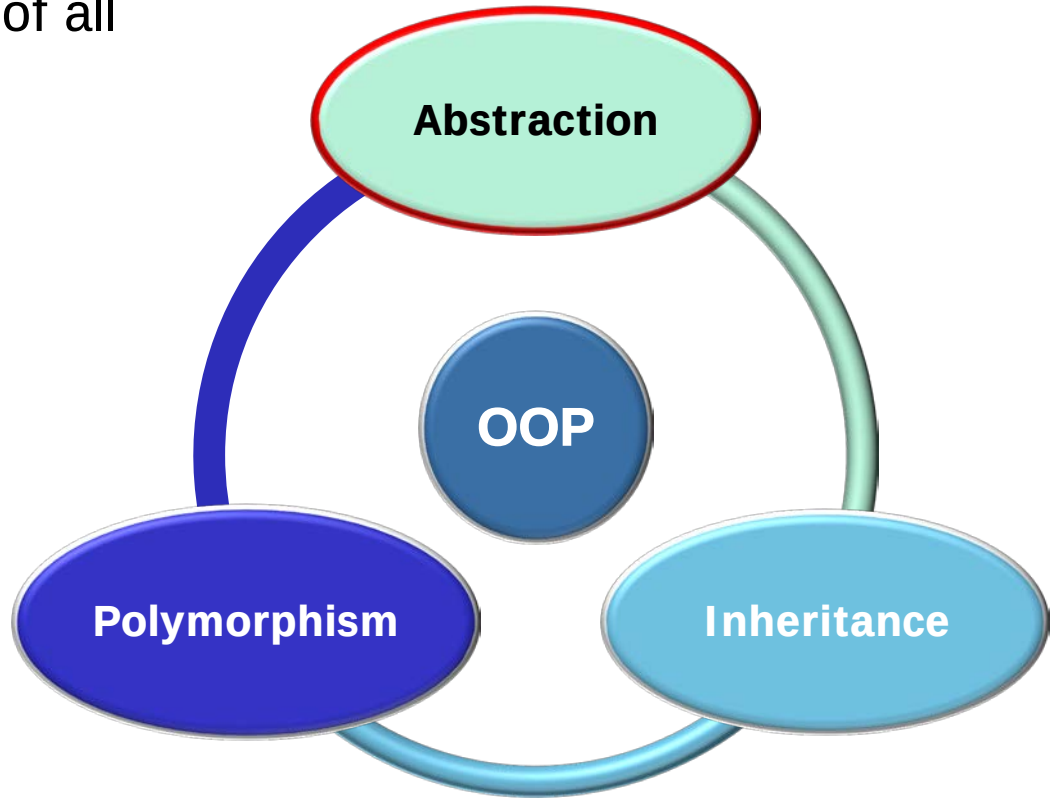


See github.com/douglascraigschmidt/CS891/tree/master/ex/ImageCounter

Overview of Java's Support for Abstraction

Overview of Java's Support for Abstraction

- Abstraction is an essential part of all object-oriented programming languages



See en.wikipedia.org/wiki/Abstraction_(computer_science)

Overview of Java's Support for Abstraction

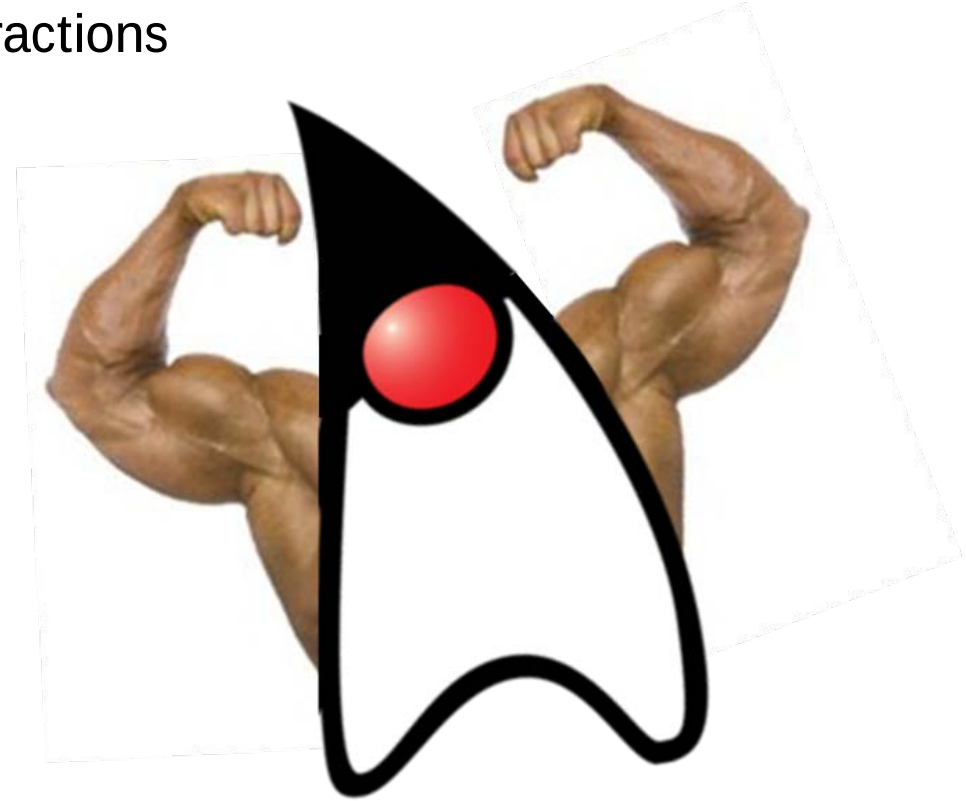
- Abstraction is an essential part of all object-oriented programming languages
  - It emphasizes what's *important* &

Overview of Java's Support for Abstraction

- Abstraction is an essential part of all object-oriented programming languages
  - It emphasizes what's *important* &
  - De-emphasizes what's *unimportant*

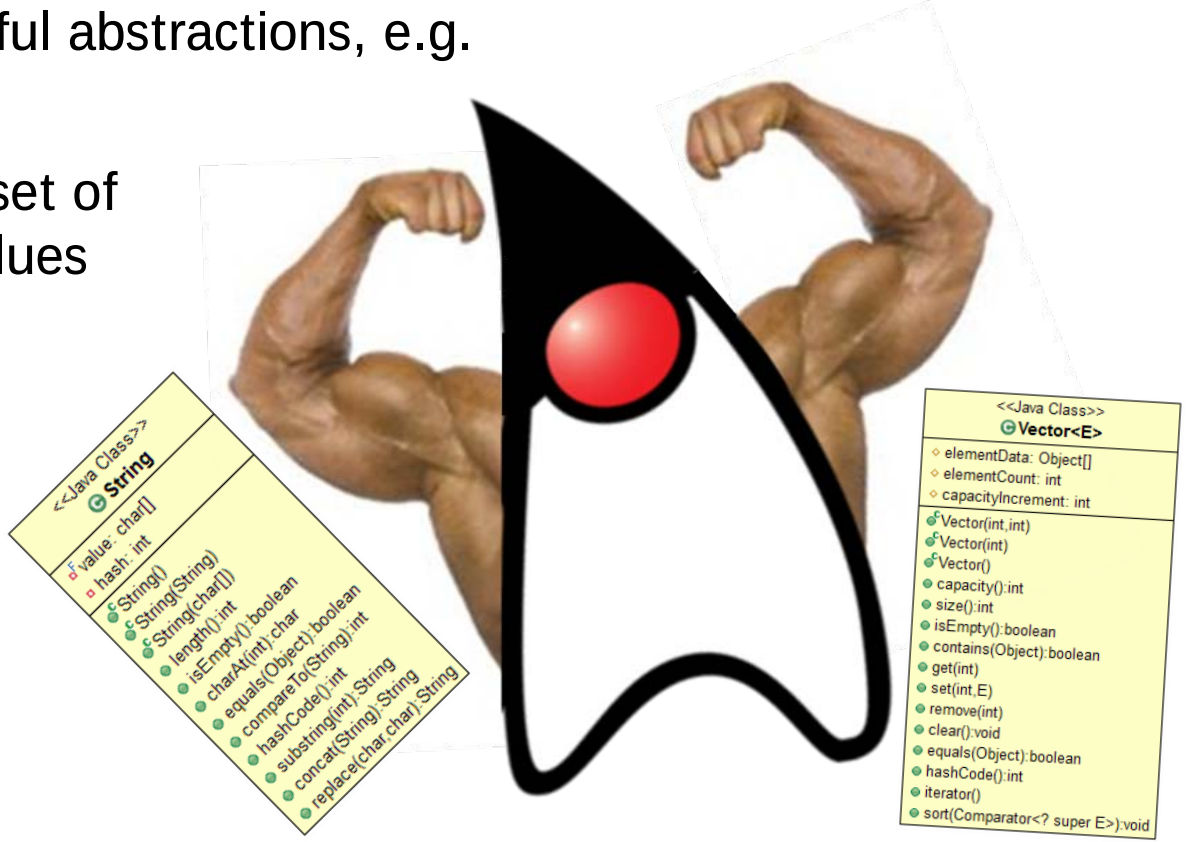
Overview of Java's Support for Abstraction

- Java supports many powerful abstractions



Overview of Java's Support for Abstraction

- Java supports many powerful abstractions, e.g.
  - Data abstractions
    - e.g., a set of values & set of operations on those values



See en.wikipedia.org/wiki/Abstraction_(computer_science)#Data_abstraction

Overview of Java's Support for Abstraction

- Java supports many powerful abstractions, e.g.
  - Data abstractions
    - e.g., a set of values & set of operations on those values

Data should only be accessed via APIs

See en.wikipedia.org/wiki/Application_programming_interface

Overview of Java's Support for Abstraction

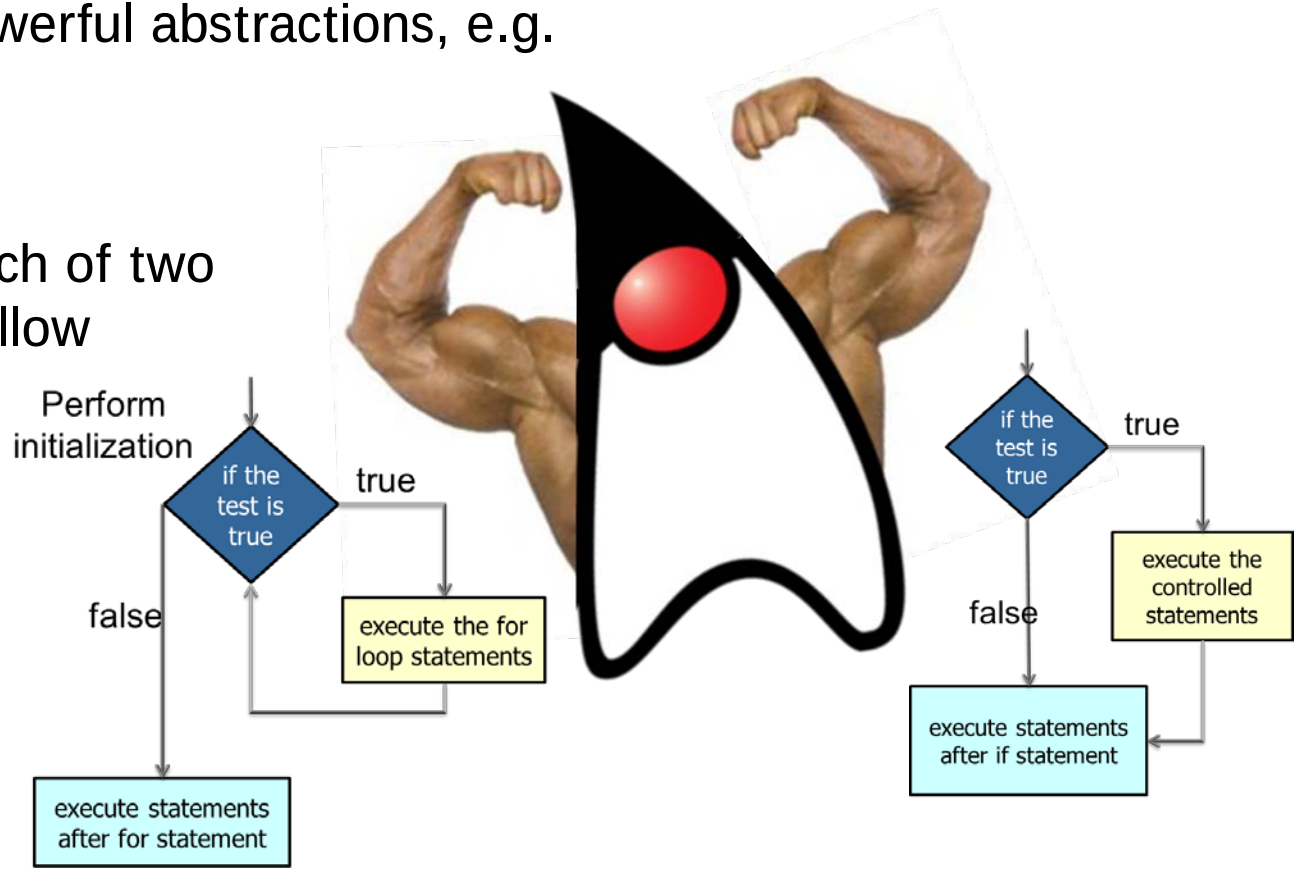
- Java supports many powerful abstractions, e.g.
  - Data abstractions
    - e.g., a set of values & set of operations on those values

Data should only be accessed via APIs

The goal is to avoid "breaking" code when inevitable changes to data occur..

Overview of Java's Support for Abstraction

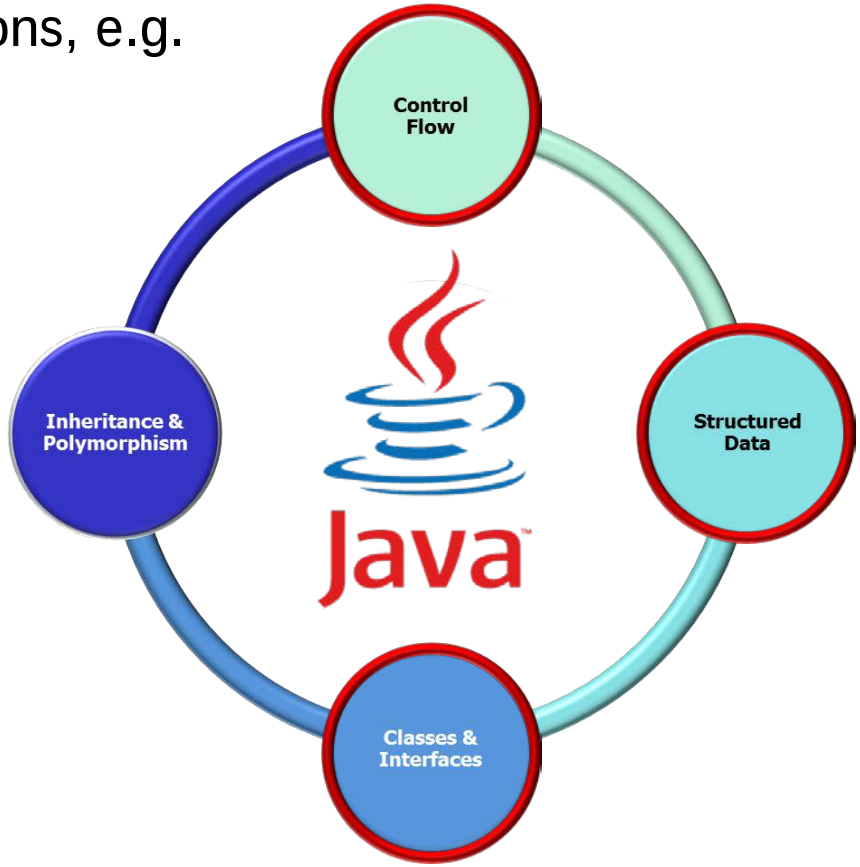
- Java supports many powerful abstractions, e.g.
  - Data abstractions
  - Control abstractions
    - e.g., determine which of two or more paths to follow



See en.wikipedia.org/wiki/Abstraction_(computer_science)#Control_abstraction

Overview of Java's Support for Abstraction

- Java supports many powerful abstractions, e.g.
  - Data abstractions
  - Control abstractions

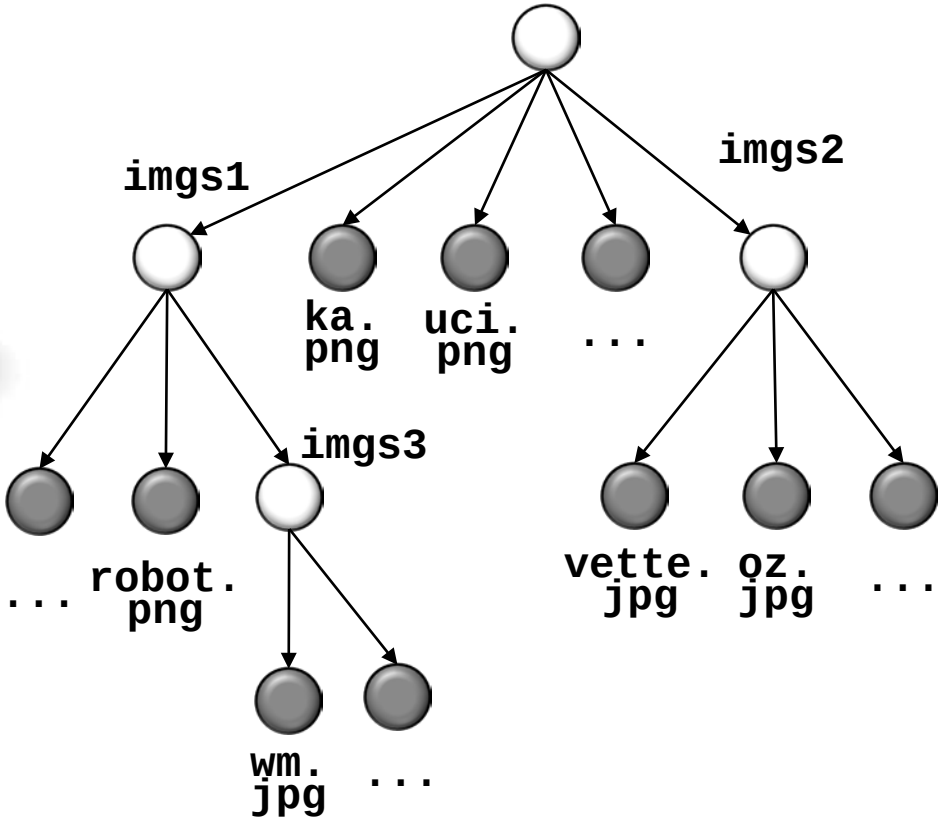


We'll now summarize various data & control abstractions supported by Java

Overview of the Image Counter Example

Overview of the Image Counter Example

- We show how to apply key Java abstractions in the context of a program that crawls web pages recursively

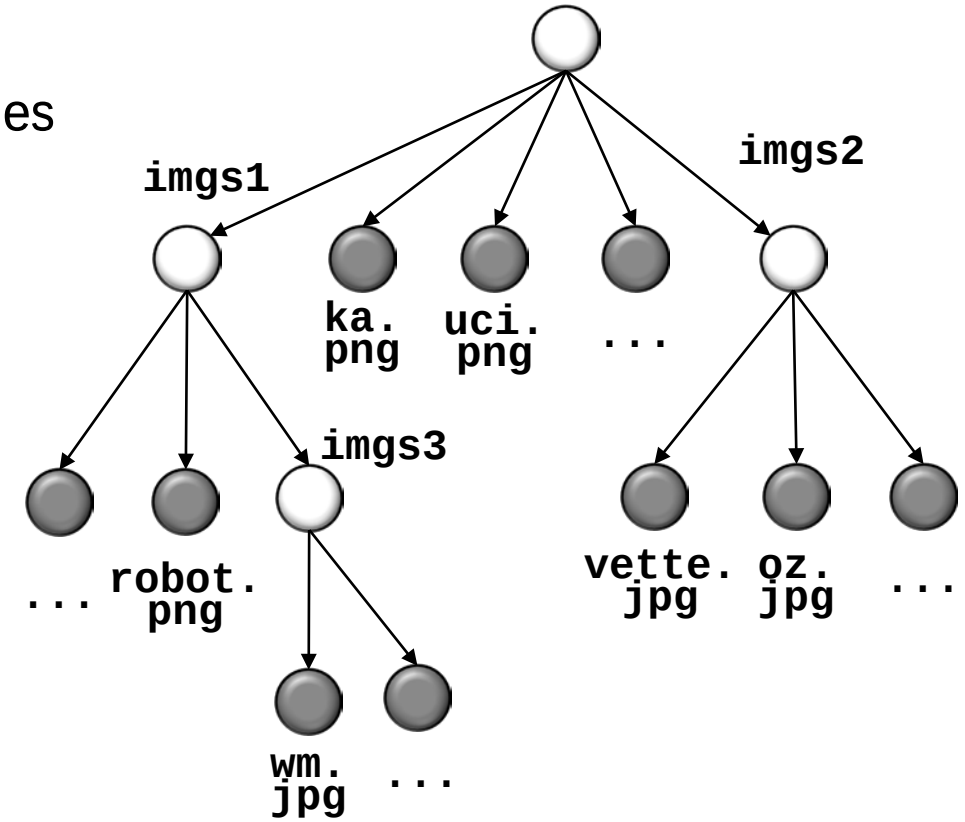


See github.com/douglascraigschmidt/CS891/tree/master/ex/ImageCounter

Overview of the Image Counter Example

- We show how to apply key Java abstractions in the context of a program that crawls web pages recursively
  - This program counts the # of images on each page

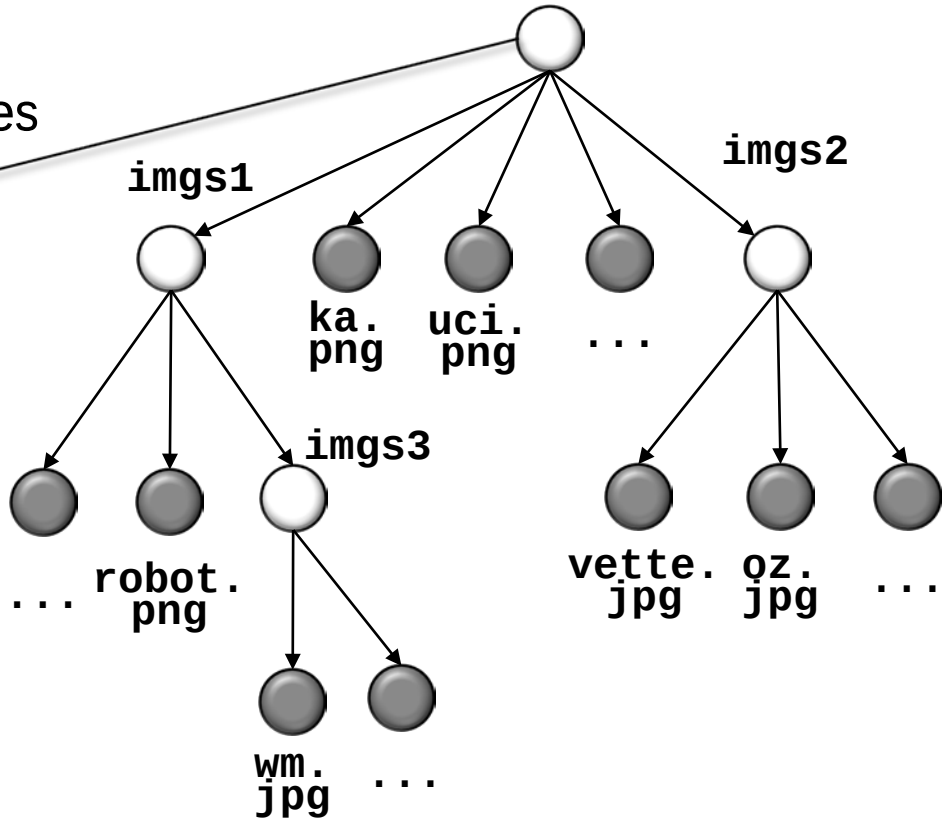
```
>> Depth: 1 [index.html](1)
>> Depth: 2 [imgs1/index.html](12)
>> Depth: 2 [imgs1/index.html](12)
Already processed imgs1/index.html
>> Depth: 2 [imgs2/index.html](12)
>> Depth: 3 [imgs3/index.html](13)
Exceeded max depth of 2
there are 21 total image(s)
reachable from index.html
```



Overview of the Image Counter Example

- We show how to apply key Java abstractions in the context of a program that crawls web pages recursively
  - This program counts the # of images on each page

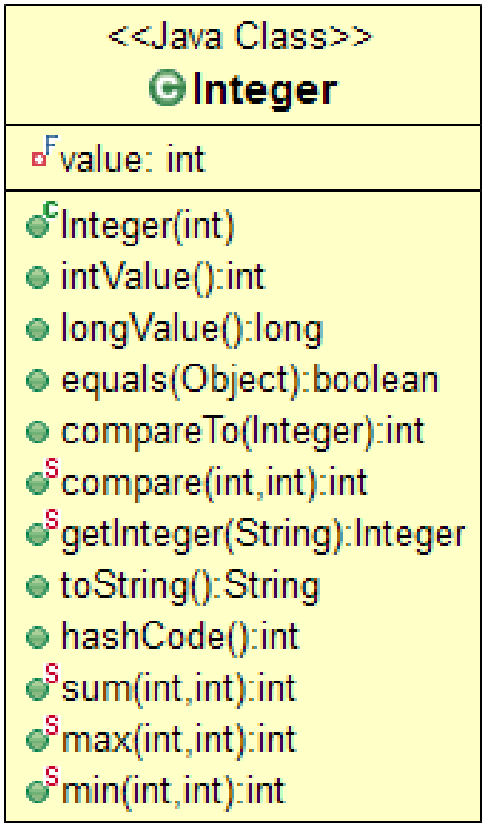
The root folder can either reside locally (filesystem-based) or be accessed remotely (web-based)



Overview of Java's Support for Data Abstraction

Overview of Java's Support for Data Abstraction

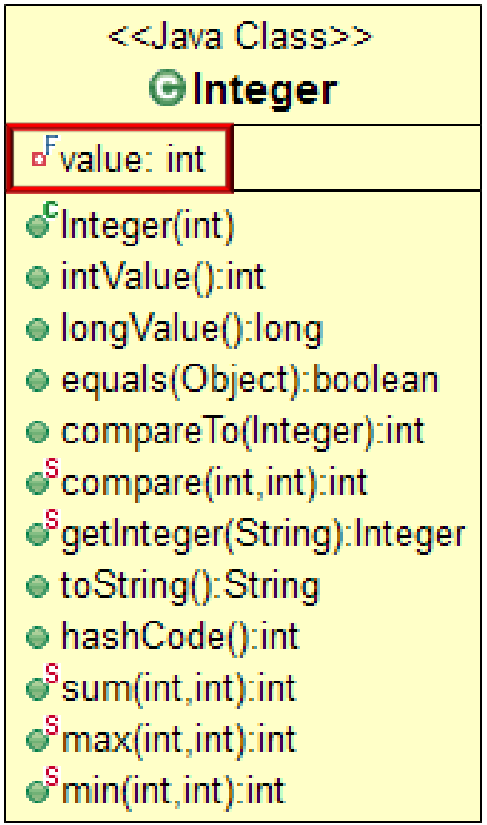
- Java supports data abstraction via Abstract Data Types (ADTs)



See en.wikipedia.org/wiki/Abstract_data_type

Overview of Java's Support for Data Abstraction

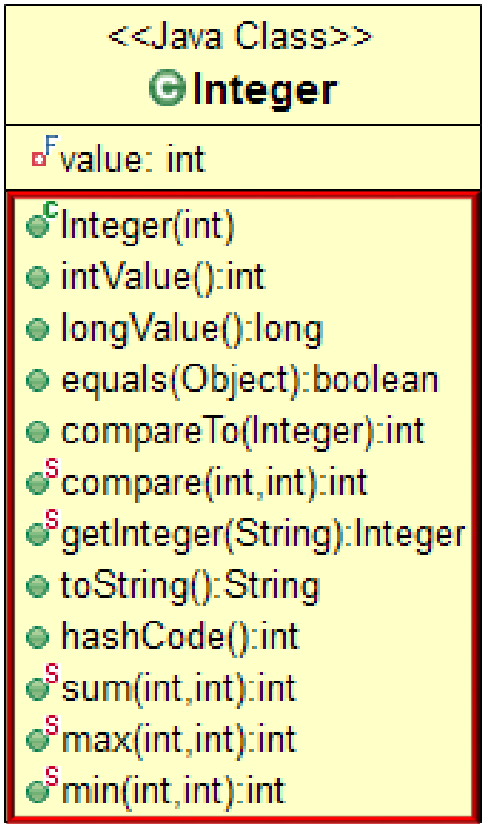
- Java supports data abstraction via Abstract Data Types (ADTs), which define
  - A set of data value(s)



See docs.oracle.com/javase/8/docs/api/java/lang/Integer.html

Overview of Java's Support for Data Abstraction

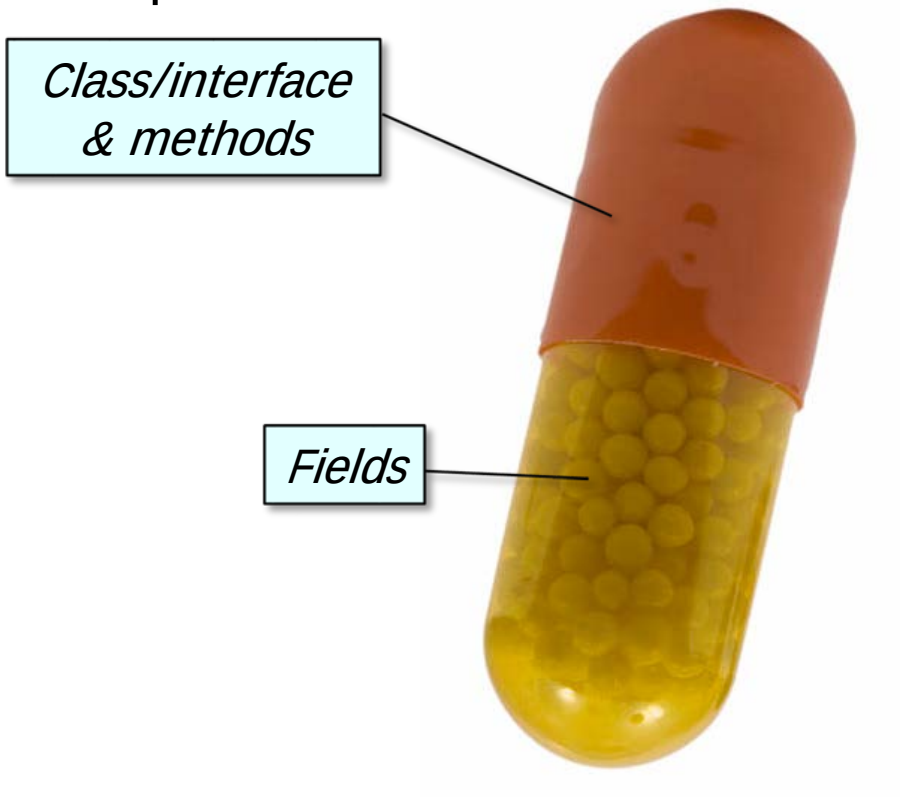
- Java supports data abstraction via Abstract Data Types (ADTs), which define
  - A set of data value(s)
  - Operations on these value(s)



See docs.oracle.com/javase/8/docs/api/java/lang/Integer.html

Overview of Java's Support for Data Abstraction

- At the heart of data abstraction is encapsulation



See en.wikipedia.org/wiki/Encapsulation_(computer_programming)

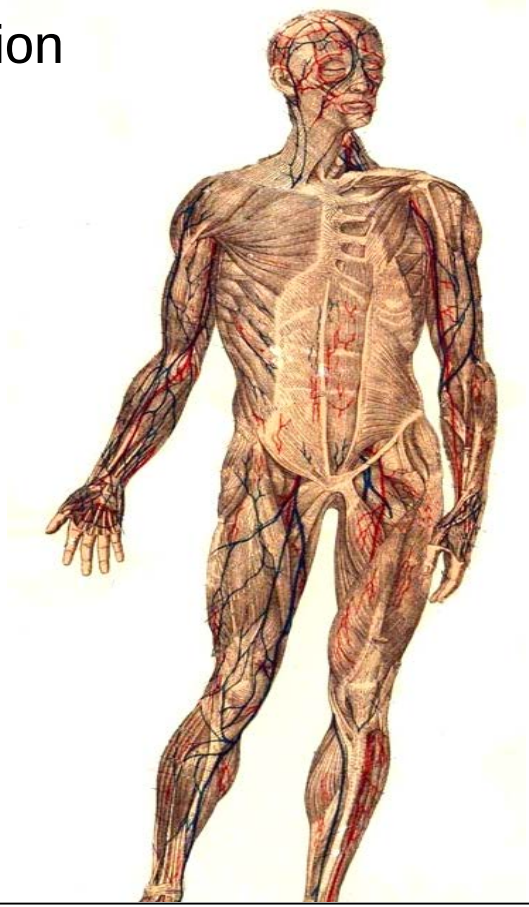
Overview of Java's Support for Data Abstraction

- At the heart of data abstraction is encapsulation
  - Hides ADT internal representation so apps can only access public operations

See en.wikipedia.org/wiki/Encapsulation_(computer_programming)

Overview of Java's Support for Data Abstraction

- At the heart of data abstraction is encapsulation
  - Hides ADT internal representation so apps can only access public operations
    - *Not* its implementation details



Overview of Java's Support for Data Abstraction

- At the heart of data abstraction is encapsulation
  - Hides ADT internal representation so apps can only access public operations
    - *Not* its implementation details

Encapsulation protects a program from internal implementation changes

Overview of Java's Support for Classes & Interfaces

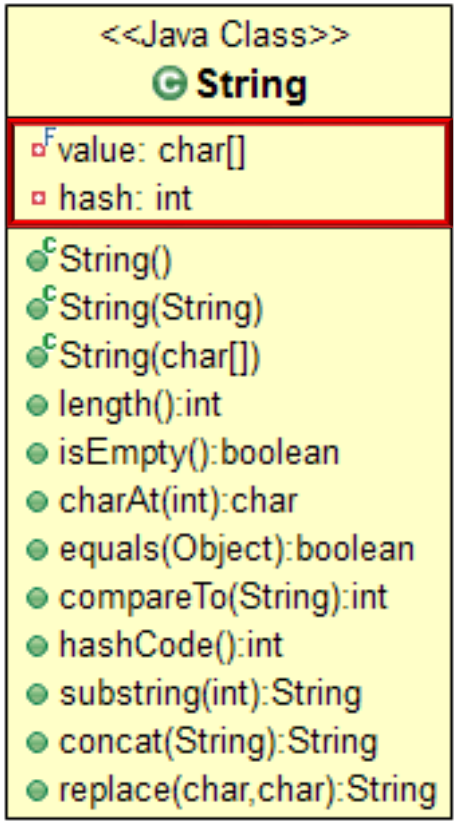
Overview of Java's Support for Classes & Interfaces

- Java classes provide a blueprint for creating objects

See docs.oracle.com/javase/tutorial/java/javaOO/classes.html

Overview of Java's Support for Classes & Interfaces

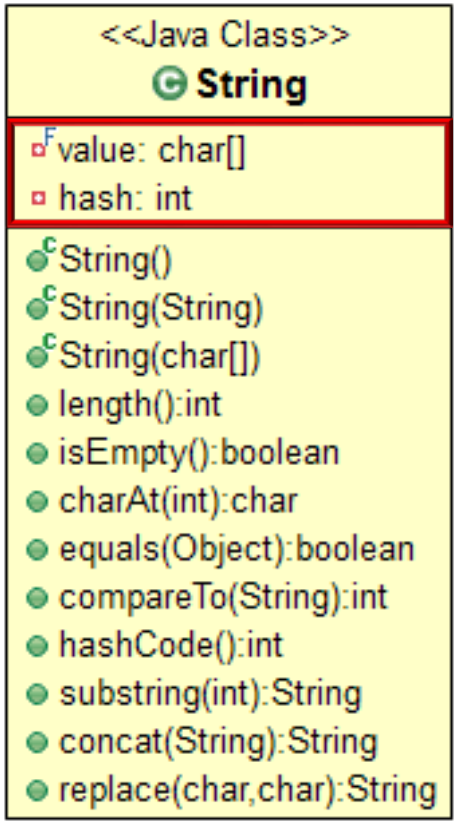
- Java classes may contain several entities
  - *Fields*
    - Used to store the state of an object



See docs.oracle.com/javase/8/docs/api/java/lang/String.html

Overview of Java's Support for Classes & Interfaces

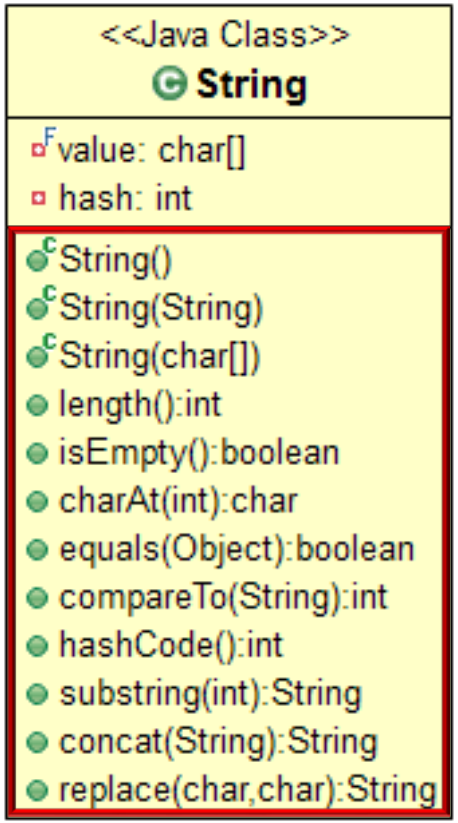
- Java classes may contain several entities
  - *Fields*
    - Used to store the state of an object
      - e.g., store a sequence of characters, length of this sequence, etc.



See docs.oracle.com/javase/8/docs/api/java/lang/String.html

Overview of Java's Support for Classes & Interfaces

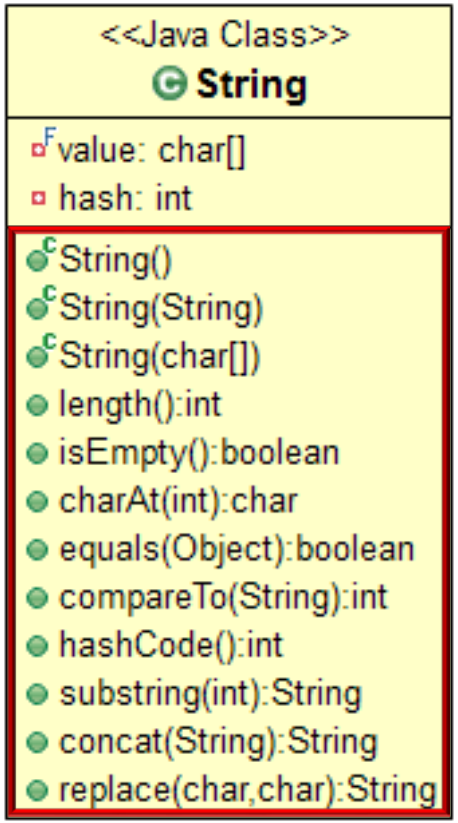
- Java classes may contain several entities
  - *Fields*
  - *Methods*
    - Used to implement the behaviors of an object



See docs.oracle.com/javase/8/docs/api/java/lang/String.html

Overview of Java's Support for Classes & Interfaces

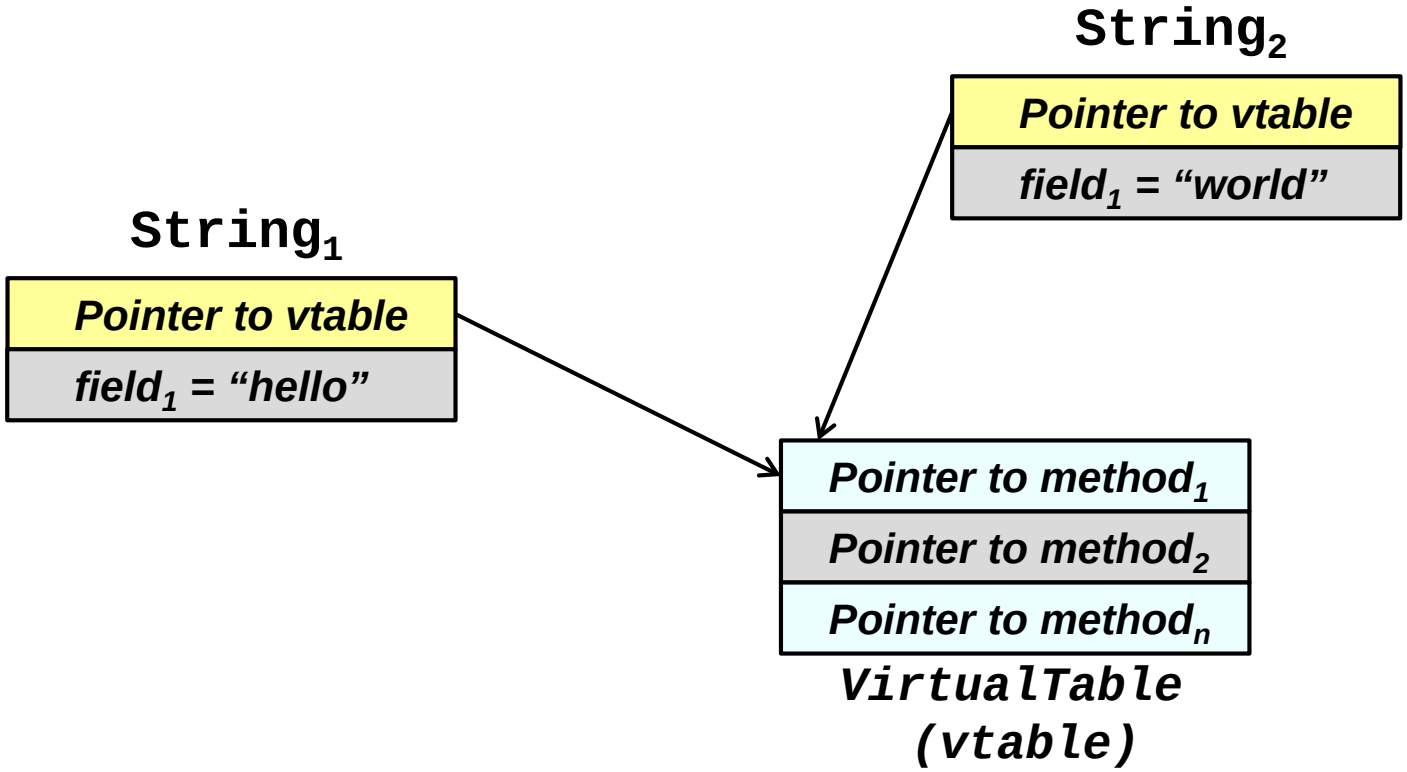
- Java classes may contain several entities
  - *Fields*
  - *Methods*
    - Used to implement the behaviors of an object
      - e.g., examine individual characters of the sequence, compare strings, search strings, extract substrings, create copies of a string, etc.



See docs.oracle.com/javase/8/docs/api/java/lang/String.html

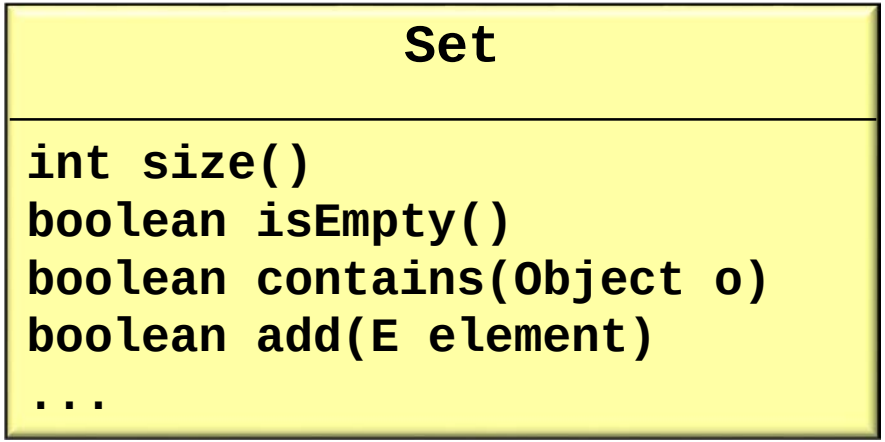
Overview of Java's Support for Classes & Interfaces

- Objects of same class share methods, but may store different values in fields



Overview of Java's Support for Classes & Interfaces

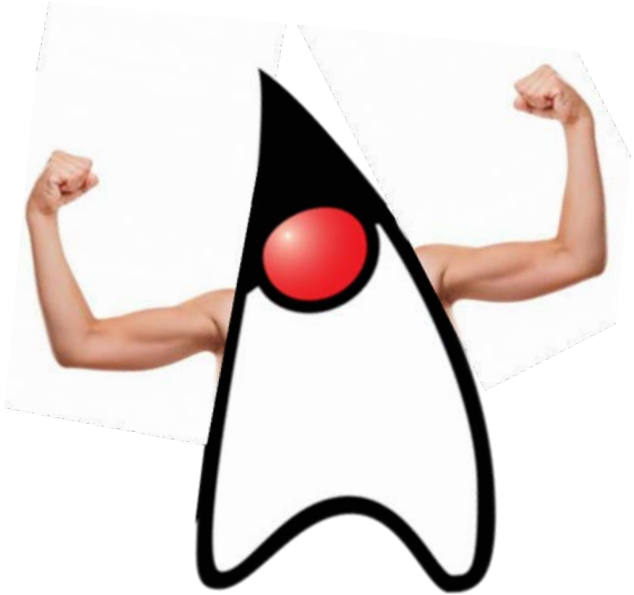
- Java interfaces define a contract specifying methods that classes implementing the interface provide



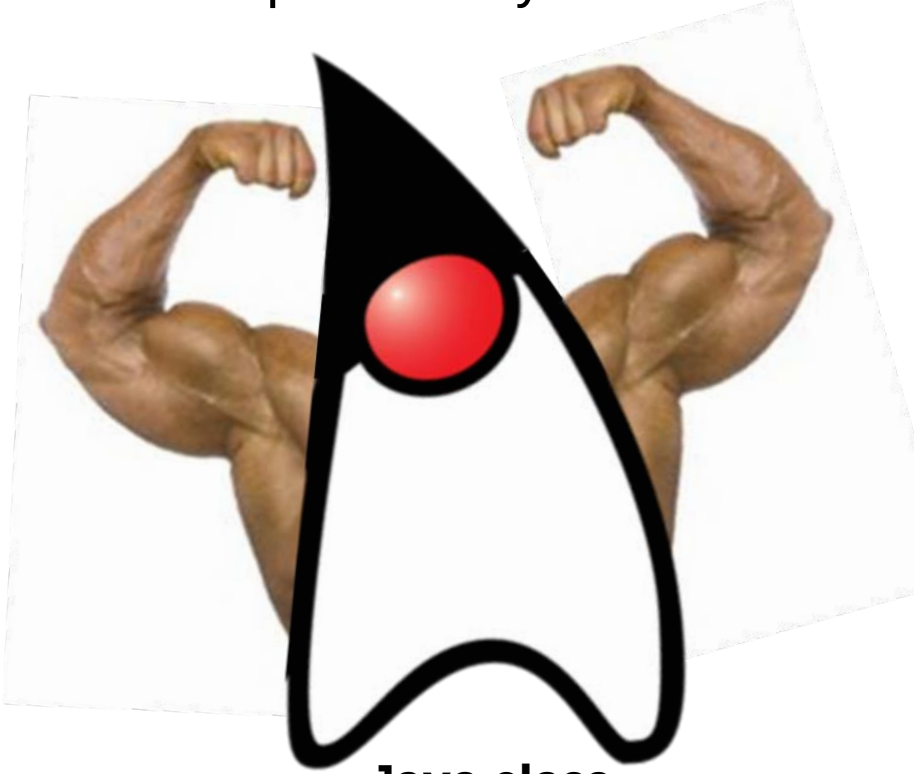
See docs.oracle.com/javase/tutorial/java/concepts/interface.html

Overview of Java's Support for Classes & Interfaces

- A Java interfaces provides a subset of the features provided by a Java class



Java interface

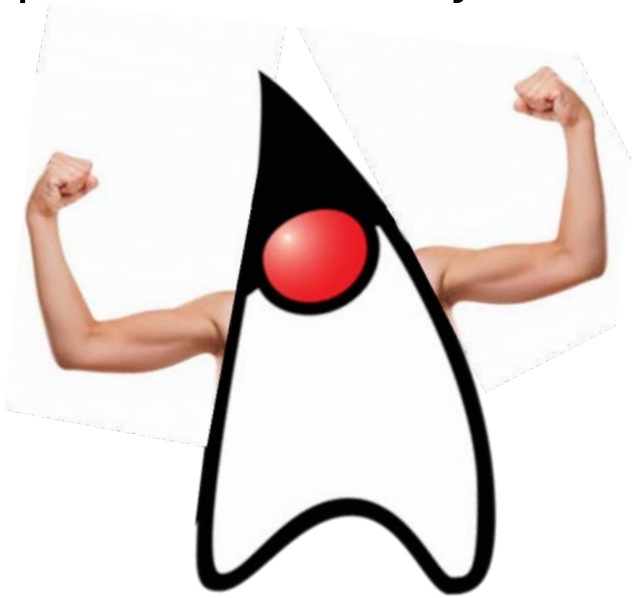


Java class

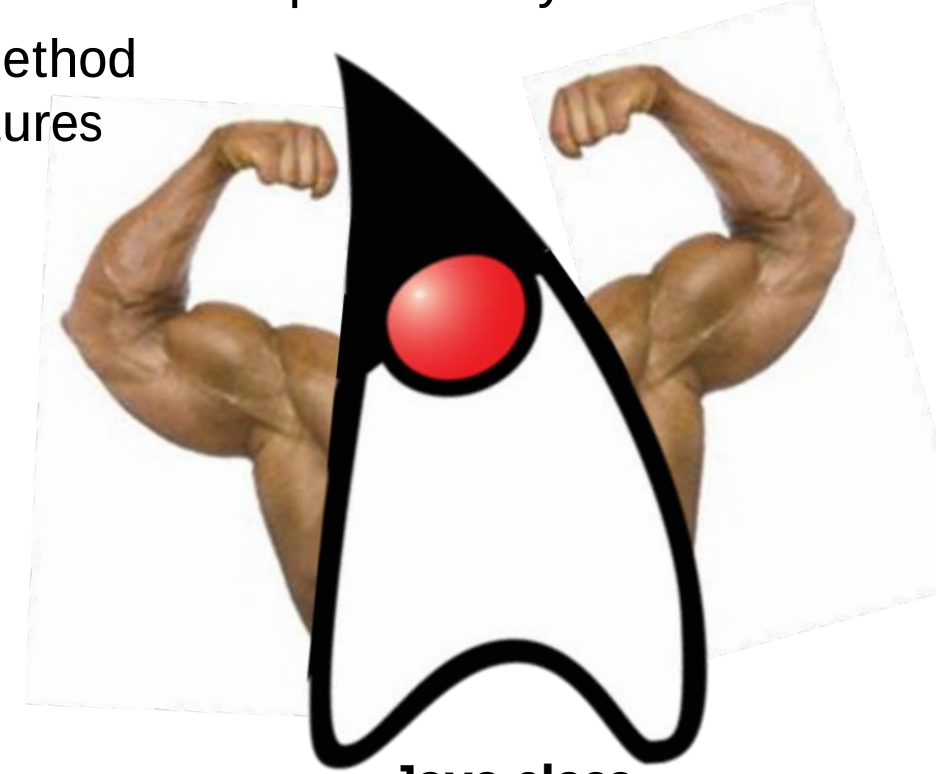
See www.tutorialspoint.com/java/java_interfaces.htm

Overview of Java's Support for Classes & Interfaces

- A Java interfaces provides a subset of the features provided by a Java class
  - e.g., Java <= 7 interfaces have no method implementations, only method signatures



Java interface

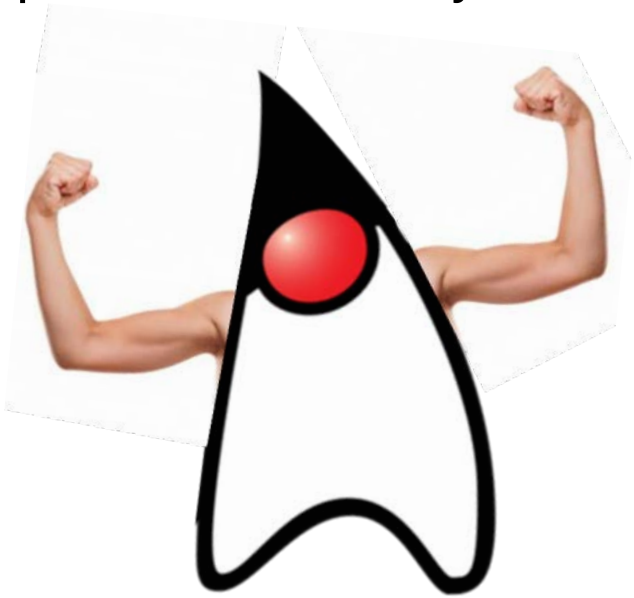


Java class

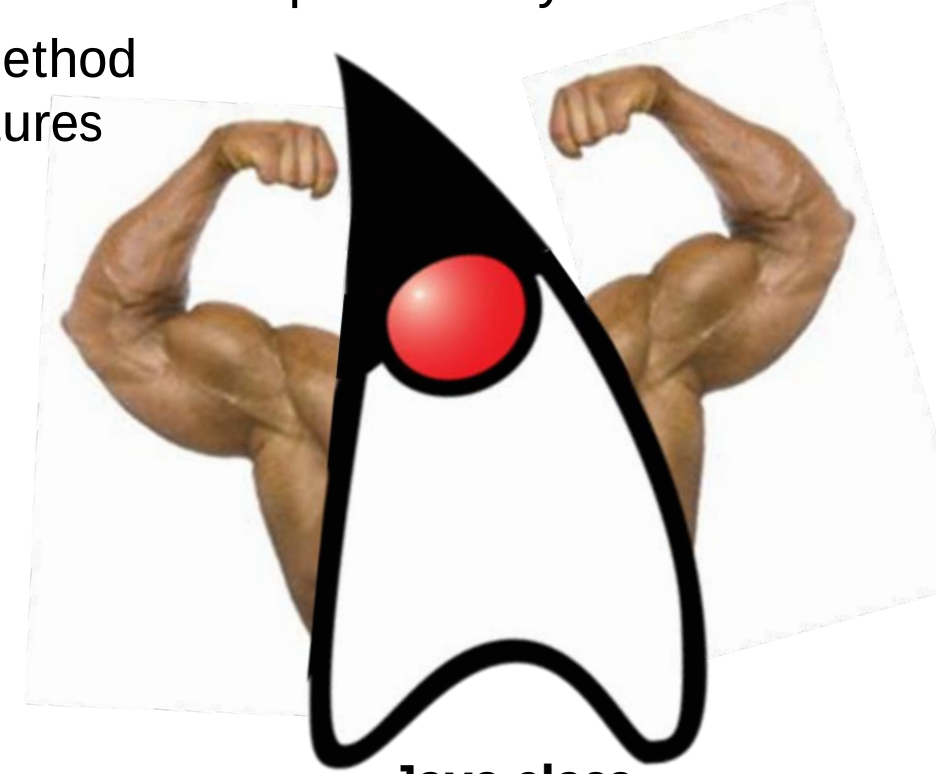
See en.wikipedia.org/wiki/Type_signature#Java_2

Overview of Java's Support for Classes & Interfaces

- A Java interfaces provides a subset of the features provided by a Java class
  - e.g., Java <= 7 interfaces have no method implementations, only method signatures



Java interface

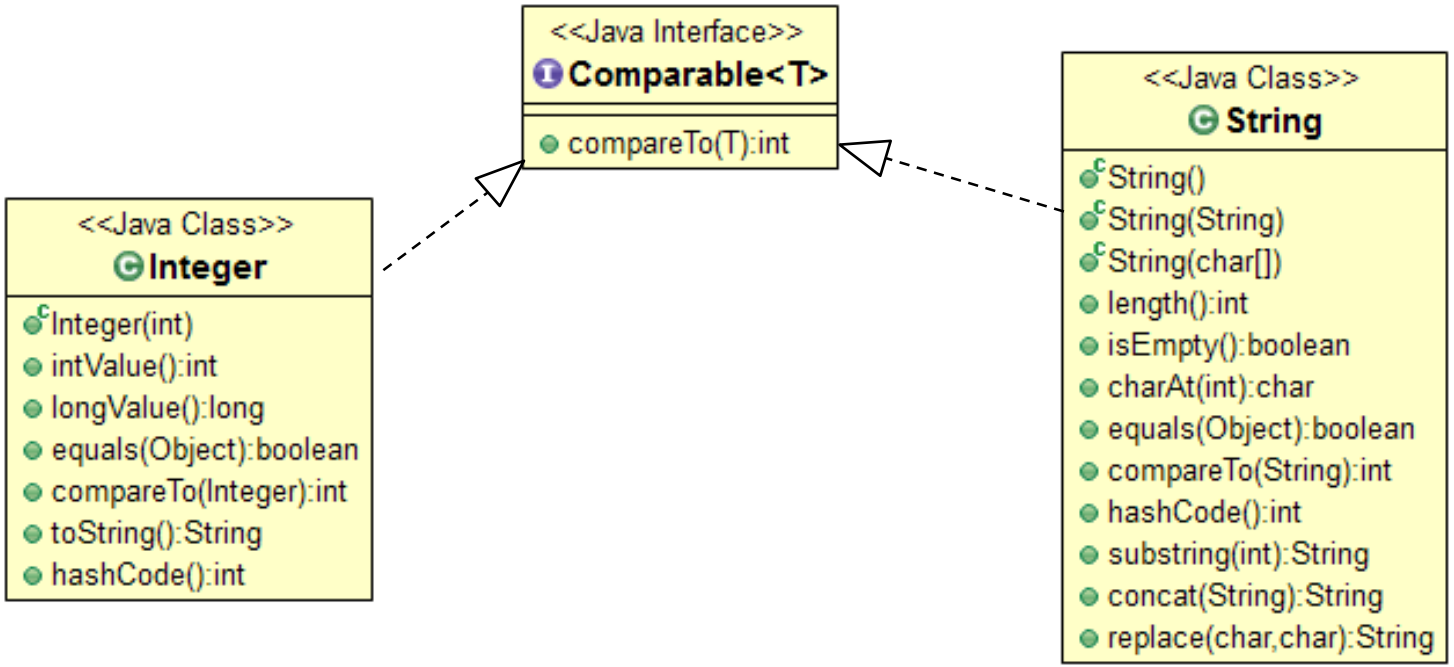


Java class

Java 8 supports default & static methods in interfaces

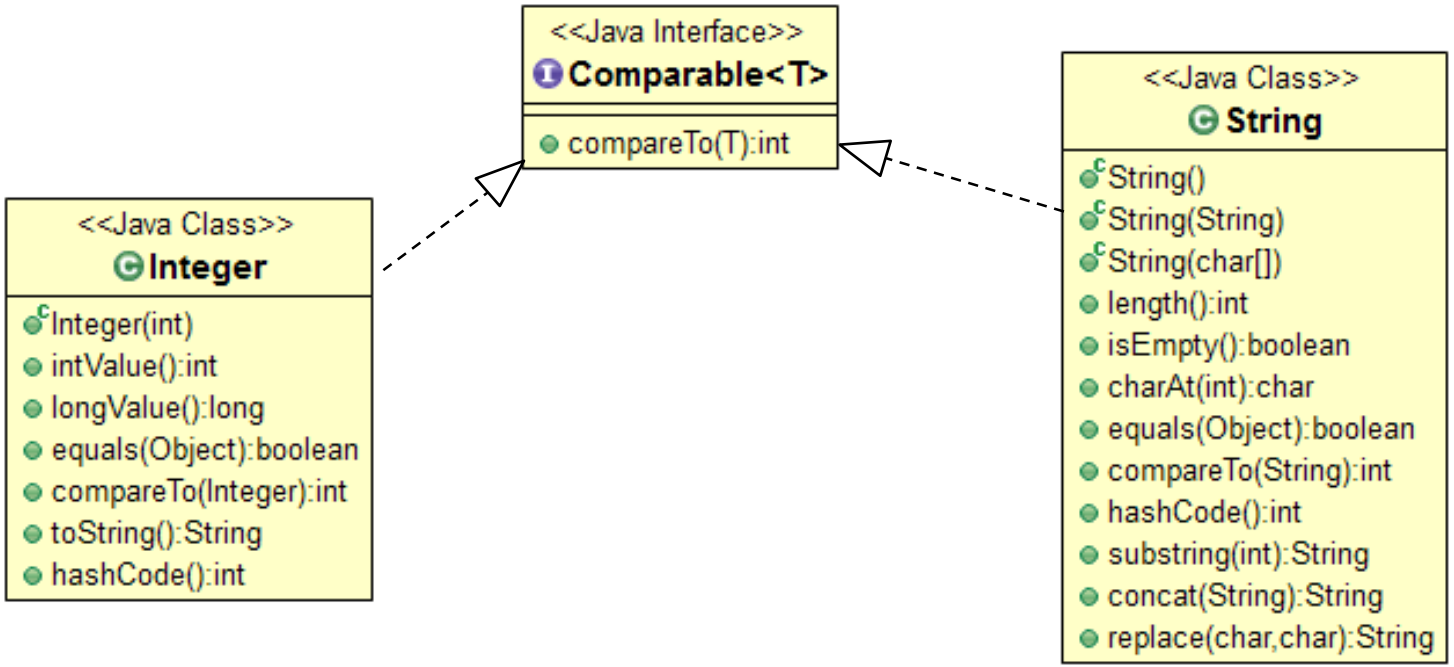
Overview of Java's Support for Classes & Interfaces

- A Java interface cannot be instantiated, but must be implemented by a class



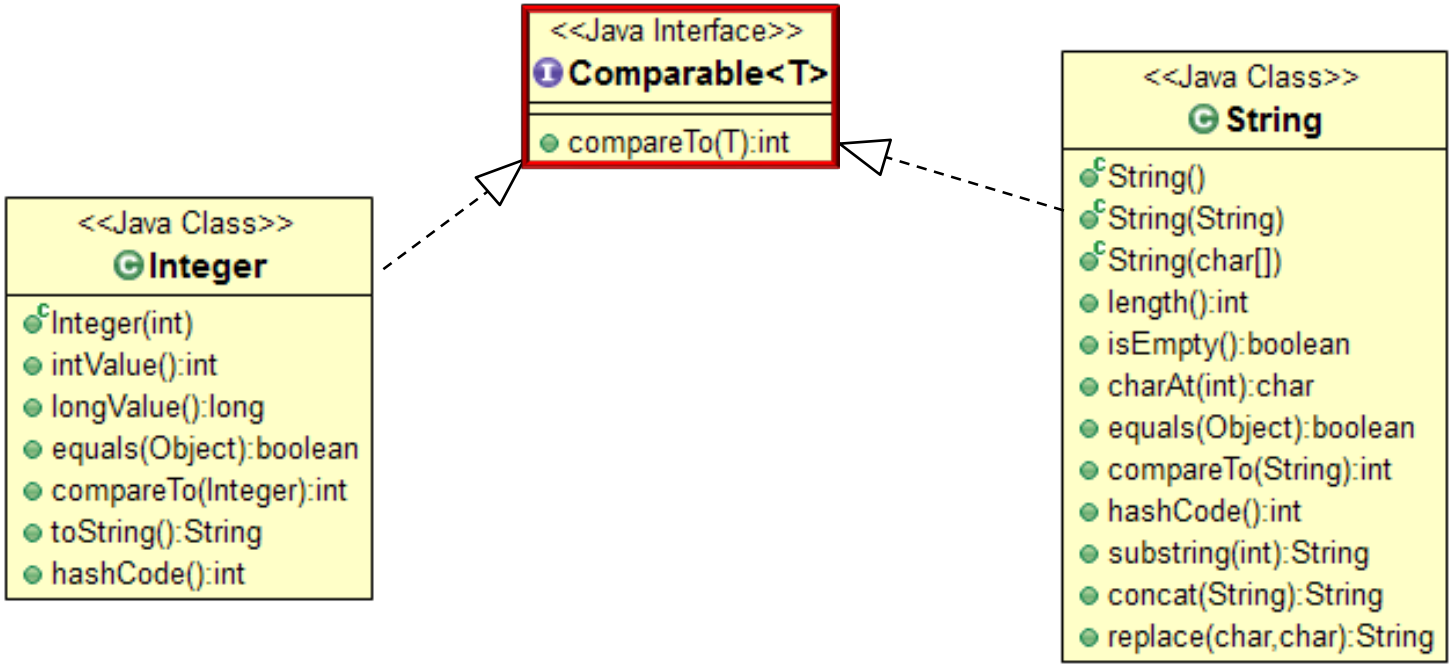
Overview of Java's Support for Classes & Interfaces

- A Java interface cannot be instantiated, but must be implemented by a class
  - The class defines the interfaces methods & any necessary fields



Overview of Java's Support for Classes & Interfaces

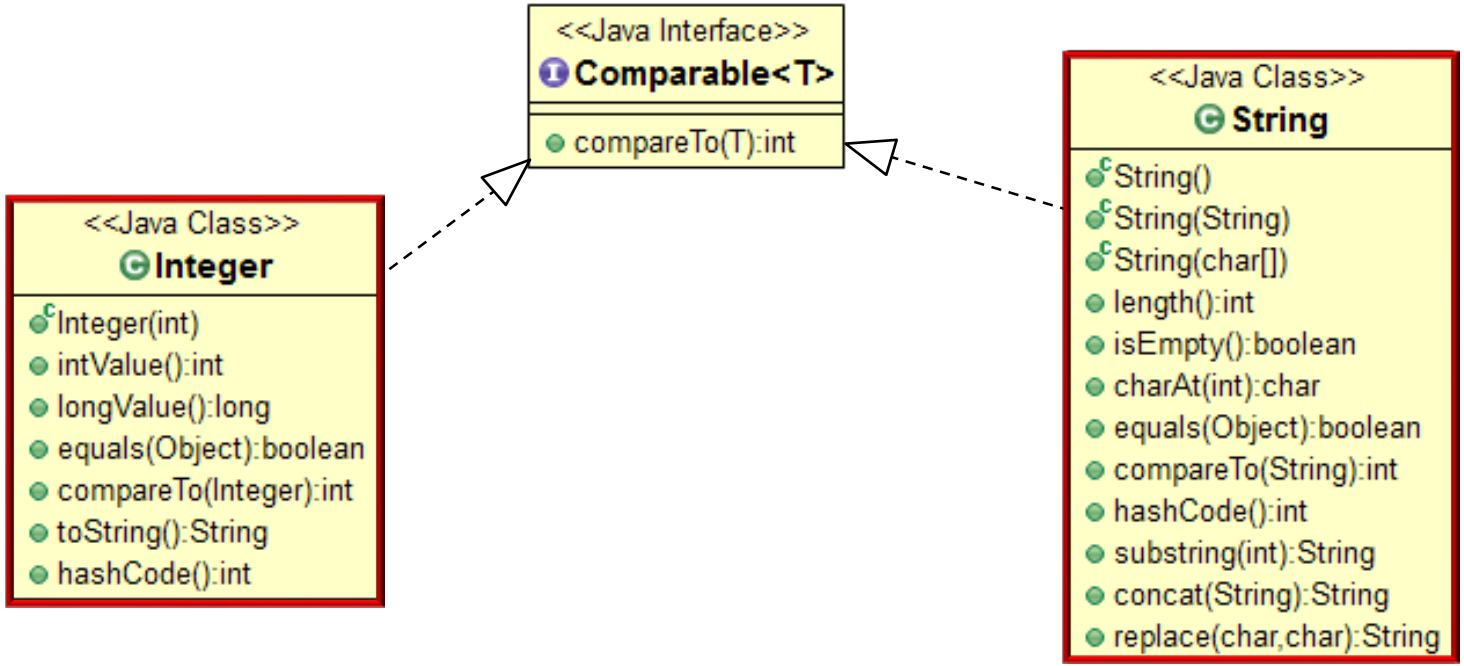
- A Java interface cannot be instantiated, but must be implemented by a class
  - The class defines the interfaces methods & any necessary fields



See docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html

Overview of Java's Support for Classes & Interfaces

- A Java interface cannot be instantiated, but must be implemented by a class
  - The class defines the interfaces methods & any necessary fields

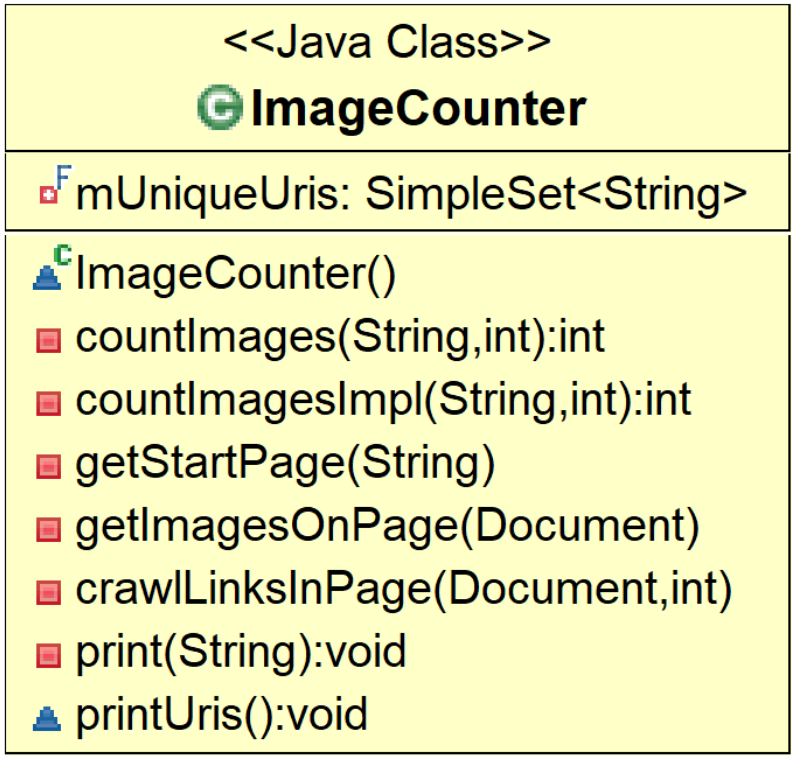


See docs.oracle.com/javase/8/docs/api/java/lang/Comparable.html

Image Counter Example of Java Classes

Image Counter Example of Java Classes

- ImageCounter uses Java's class data abstraction to count the # of images in a recursive folder structure



See ImageCounter/src/main/java/ImageCounter.java

Image Counter Example of Java Classes

- ImageCounter uses Java's class data abstraction to count the # of images in a recursive folder structure
  - Its SimpleSet field caches Uris so they are only visited once

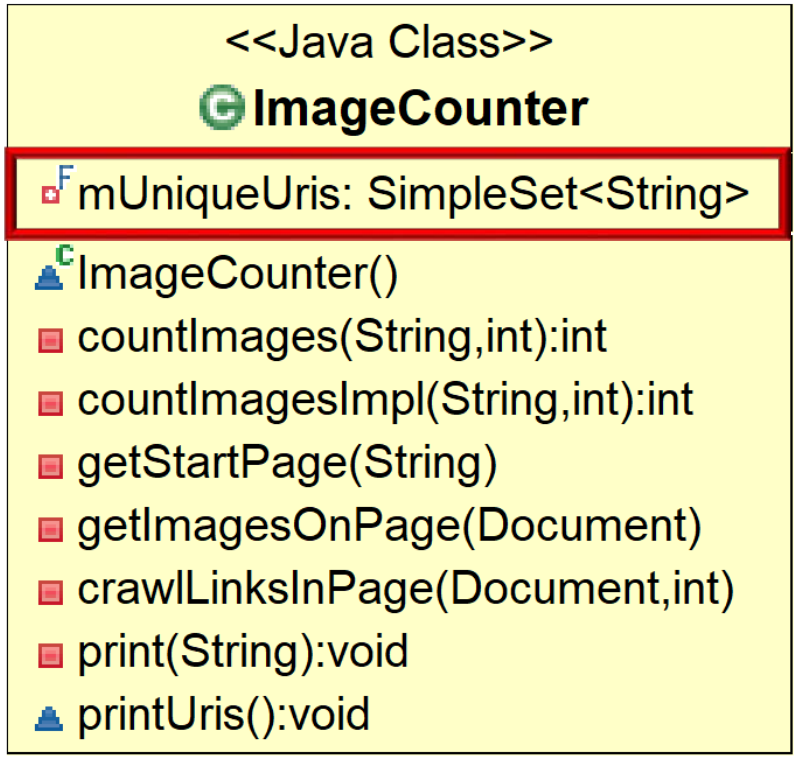


Image Counter Example of Java Classes

- ImageCounter uses Java's class data abstraction to count the # of images in a recursive folder structure
  - Its SimpleSet field caches Uris so they are only visited once
  - Some methods obtain & parse HTML pages

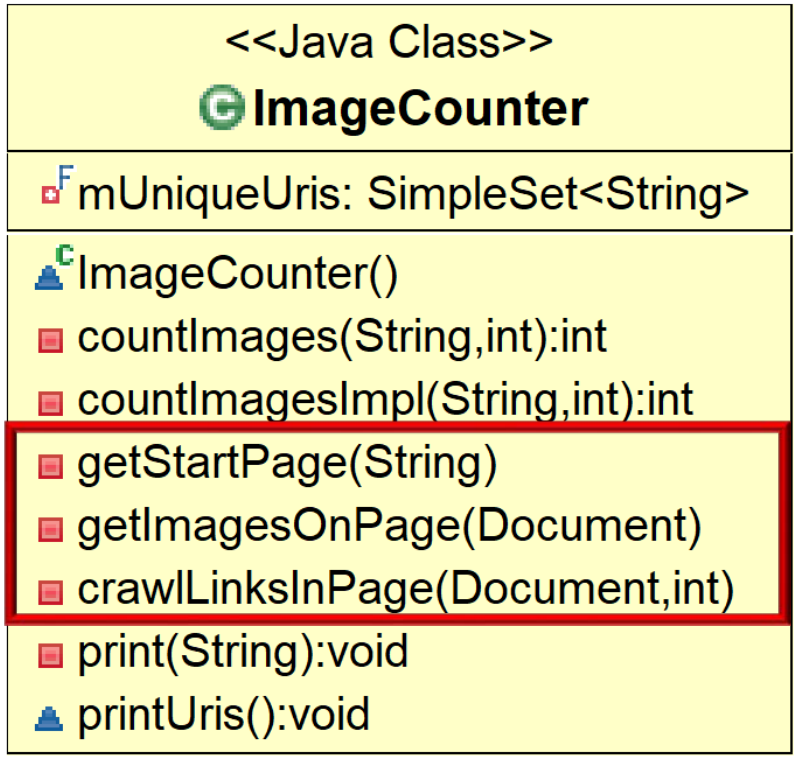


Image Counter Example of Java Classes

- ImageCounter uses Java's class data abstraction to count the # of images in a recursive folder structure
  - Its SimpleSet field caches Uris so they are only visited once
  - Some methods obtain & parse HTML pages
  - Other methods count # of images on a page (recursively)

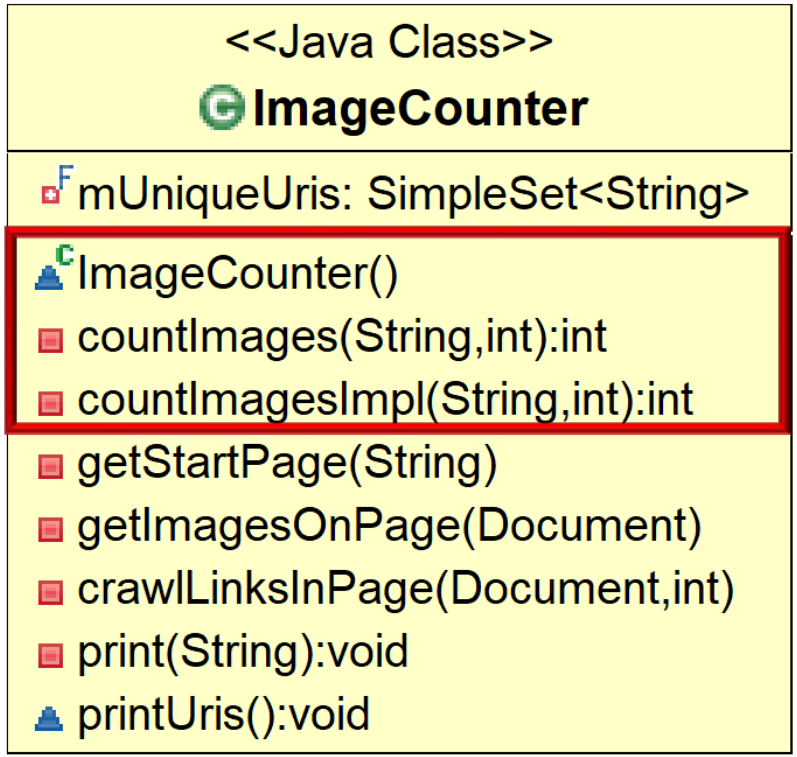
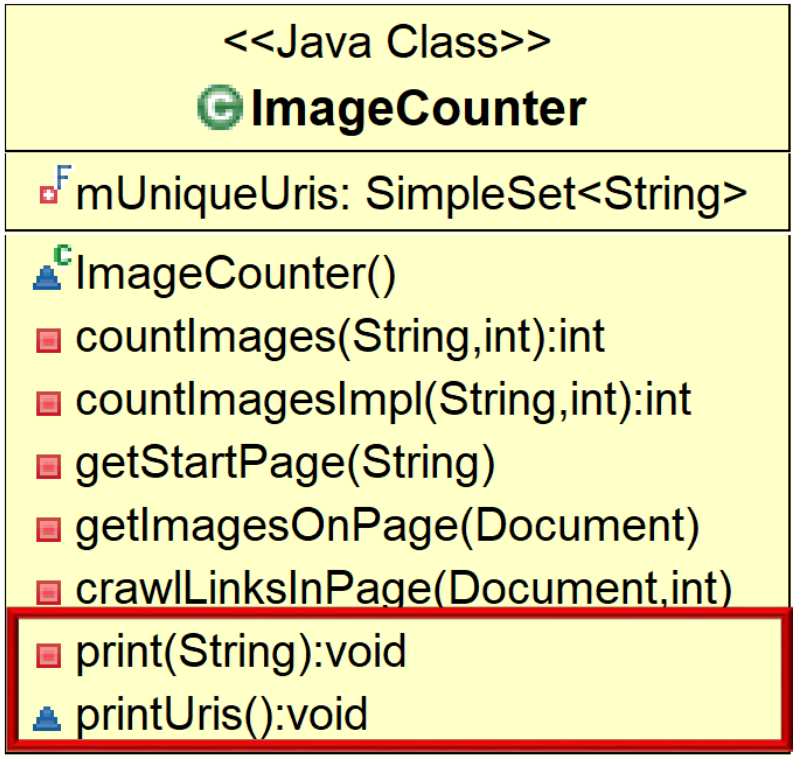


Image Counter Example of Java Classes

- ImageCounter uses Java's class data abstraction to count the # of images in a recursive folder structure
  - Its SimpleSet field caches Uris so they are only visited once
  - Some methods obtain & parse HTML pages
  - Other methods count # of images on a page (recursively)
  - Yet other methods display results to the user



Overview of Java's Support for Generics

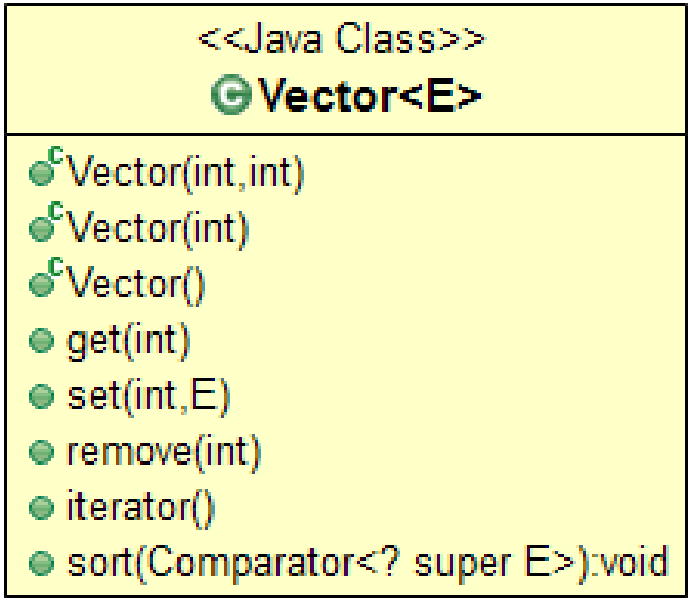
Overview of Java's Support for Generics

- Java generics enable the passing ADTs as parameters when defining classes, interfaces, & methods

See docs.oracle.com/javase/tutorial/java/generics

Overview of Java's Support for Generics

- Java generics enable the passing ADTs as parameters when defining classes, interfaces, & methods



See docs.oracle.com/javase/8/docs/api/java/util/Vector.html

Overview of Java's Support for Generics

- Java generics not identical to C++ parameterized types (templates)

Java generics

- No support for primitives
- Single copy of code exists regardless of the number of type arguments a generic code is used with
- Generified code get compiled as an entity in itself
- Bounded type parameters, possible, unbo unded defaults to Object

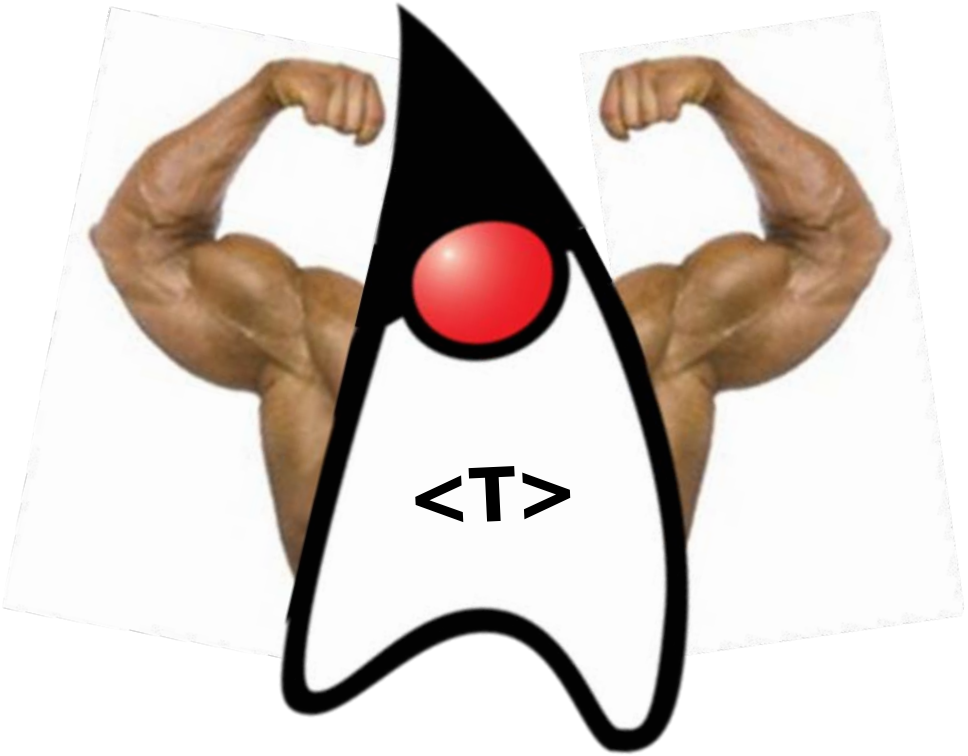
C++ templates

- Supports all types
- One copy of object code for each template instantiation
- Glorified C style marcos, compilation happens once for each expansion; some compilation errors crop up here
- No inheritance family based bounding of type parameters, only explicit specialization is possible

See www.cs.binghamton.edu/~mike/presentations/java-generics-cs580c-fall-2007.pdf

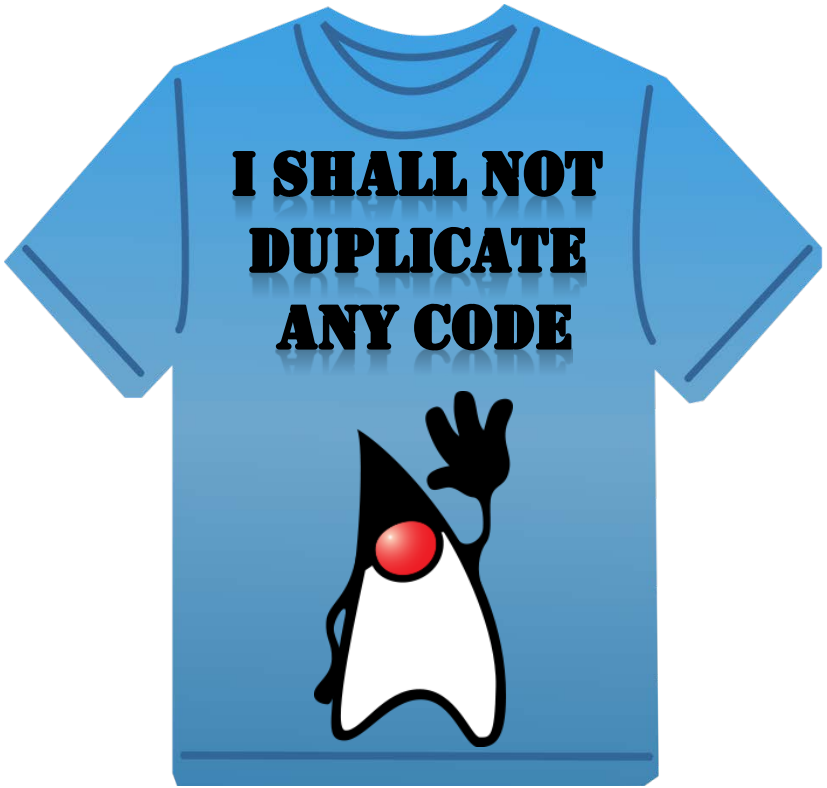
Overview of Java's Support for Generics

- Generics offer several benefits to Java programmers



Overview of Java's Support for Generics

- Generics offer several benefits to Java programmers, e.g.
  - Eliminate unnecessary code duplication



See en.wikipedia.org/wiki/Don't_repeat_yourself

Overview of Java's Support for Generics

- Generics offer several benefits to Java programmers, e.g.
  - Eliminate unnecessary code duplication
  - Ensure compile-time type safety when operating on different ADTs

```
package java.util;

public class Vector<E> ... {
   ...
```

Overview of Java's Support for Generics

- Generics offer several benefits to Java programmers, e.g.
  - Eliminate unnecessary code duplication
  - Ensure compile-time type safety when operating on different ADTs

```
package java.util;

public class Vector<E> ... {
  ...
```

e.g.,

```
Vector<Integer> vi = new
  Vector<>();
Vector<Double> vd = new
  Vector<>();

vi.set(0, 10);   // Works
vi.set(0, 10.0); // Fails
vd.set(0, 10.0); // Works
vd.set(0, 10);   // Fails
```

Overview of Java's Support for Generics

- Generics offer several benefits to Java programmers, e.g.
  - Eliminate unnecessary code duplication
  - Ensure compile-time type safety when operating on different ADTs

```
package java.util;

public class Vector<E> ... {
  ...
```

e.g.,

```
Vector<Integer> vi = new
  Vector<>();
Vector<Double> vd = new
  Vector<>();

vi.set(0, 10);   // Works
vi.set(0, 10.0); // Fails
vd.set(0, 10.0); // Works
vd.set(0, 10);   // Fails
```

Overview of Java's Support for Generics

- Generics offer several benefits to Java programmers, e.g.
  - Eliminate unnecessary code duplication
  - Ensure compile-time type safety when operating on different ADTs

```
package java.util;

public class Vector<E> ... {
  ...
```

e.g.,

```
Vector<Integer> vi = new
  Vector<>();
Vector<Double> vd = new
  Vector<>();

vi.set(0, 10);   // Works
vi.set(0, 10.0); // Fails
vd.set(0, 10.0); // Works
vd.set(0, 10);   // Fails
```

Overview of Java's Support for Generics

- Generics offer several benefits to Java programmers, e.g.
  - Eliminate unnecessary code duplication
  - Ensure compile-time type safety when operating on different ADTs

```
package java.util;

public class Vector<E> ... {
  ...
```

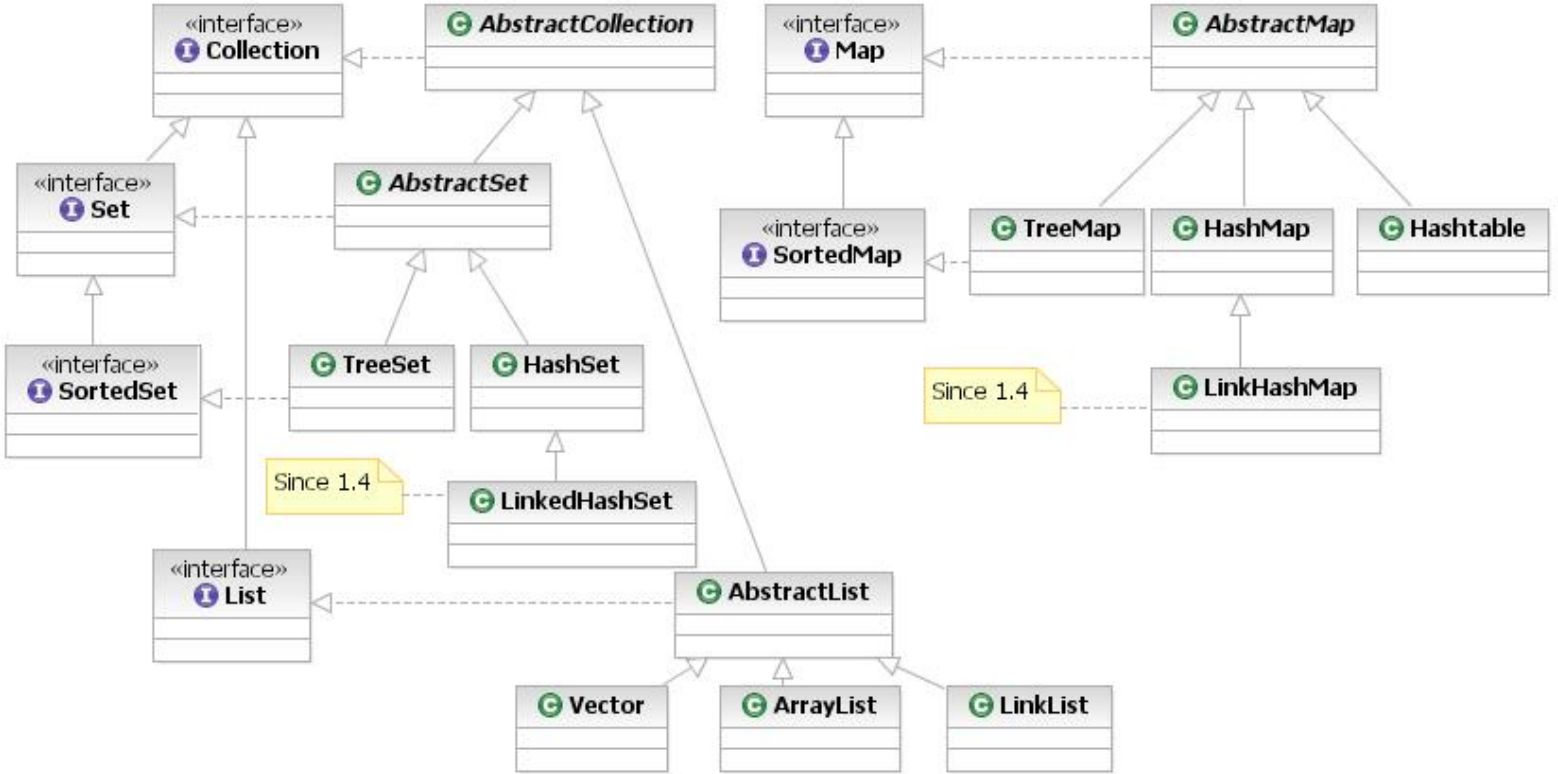
e.g.,

```
Vector<Integer> vi = new
  Vector<>();
Vector<Double> vd = new
  Vector<>();

vi.set(0, 10);   // Works
vi.set(0, 10.0); // Fails
vd.set(0, 10.0); // Works
vd.set(0, 10);   // Fails
```

Overview of Java's Support for Generics

- The Java Collections Framework uses generic classes & interfaces extensively

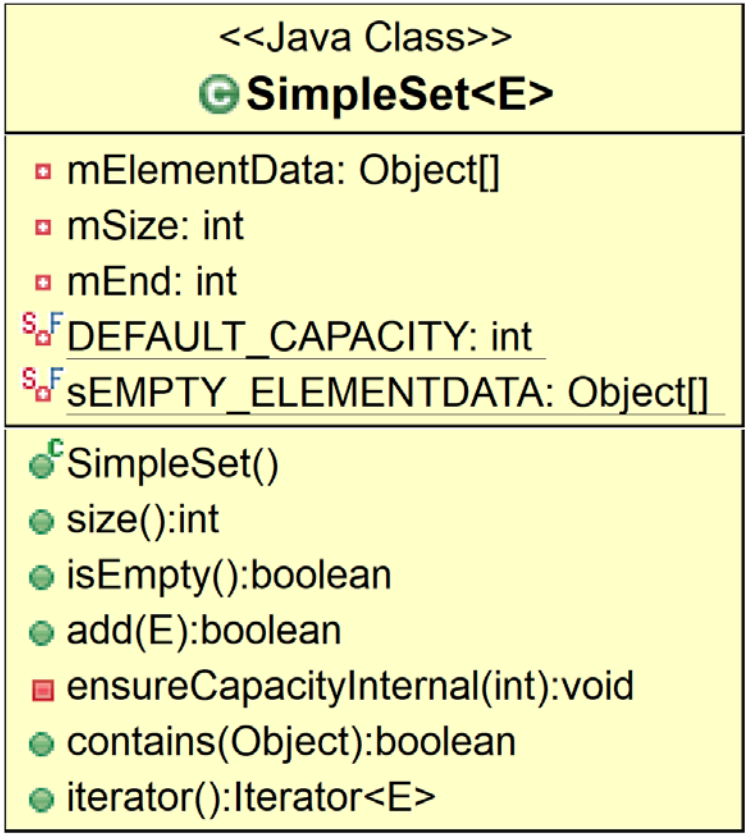


See en.wikipedia.org/wiki/Java_collections_framework

Image Counter Example of Java Generics

Image Counter Example of Java Generics

- SimpleSet defines a generic set that's implemented using a simple built-in array



See ImageCounter/src/main/java/utils/SimpleSet.java

Image Counter Example of Java Generics

- SimpleSet.add() adds the specified element to this set

```
class SimpleSet<E> ... {
  ...
  boolean add(E e) {
    if (contains(e))
      return false;

    ensureCapacityInternal
      (mSize + 1);

    mElementData[mEnd] = e;

    mEnd++;
    mSize++;
    return true;
  } ...
```

Note how the implementation of SimpleSet doesn't depend on the type of E