Overview of Key Object-Oriented Concepts Supported by Java

Key Object-Oriented Concepts Supported by Java

- Java is an object-oriented programming language

See en.wikipedia.org/wiki/Object-oriented_programming

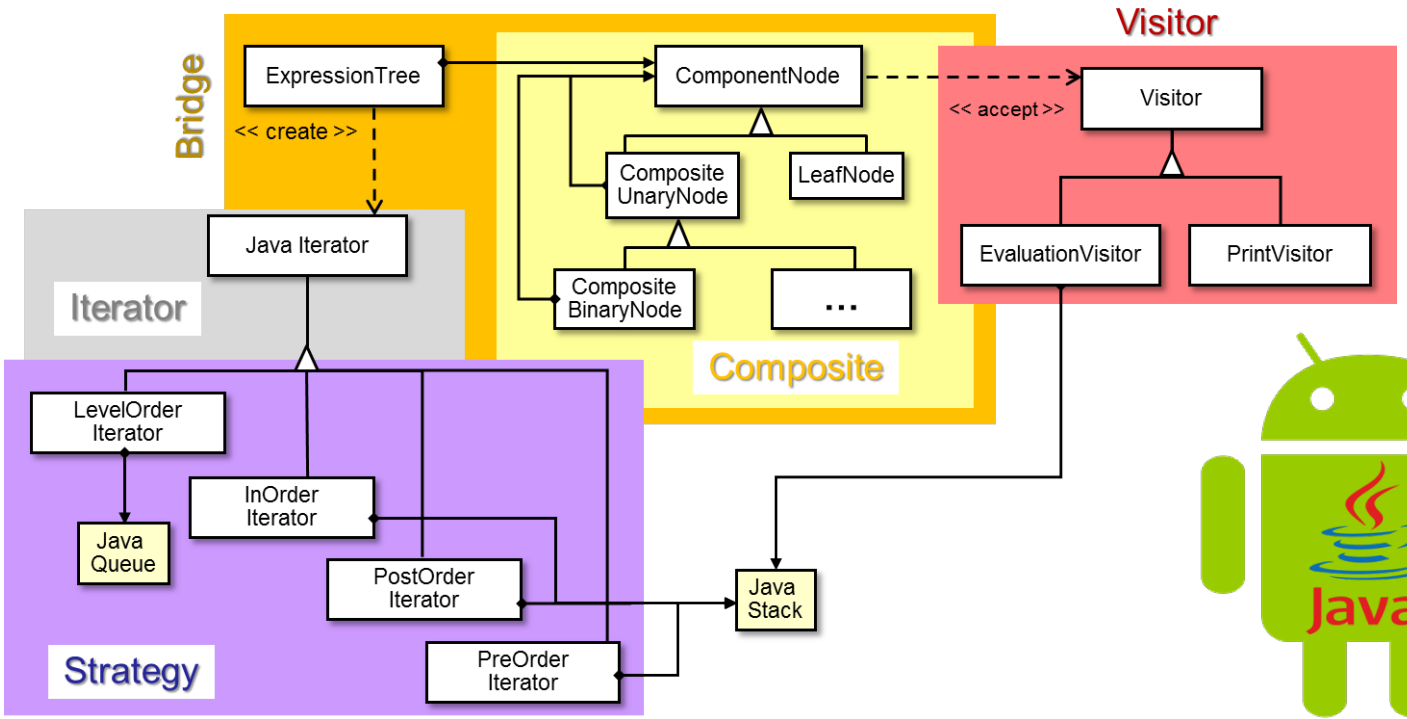
Key Object-Oriented Concepts Supported by Java

- Apps written in Java are organized in terms of *structural* elements

See en.wikipedia.org/wiki/Object-oriented_programming

Key Object-Oriented Concepts Supported by Java

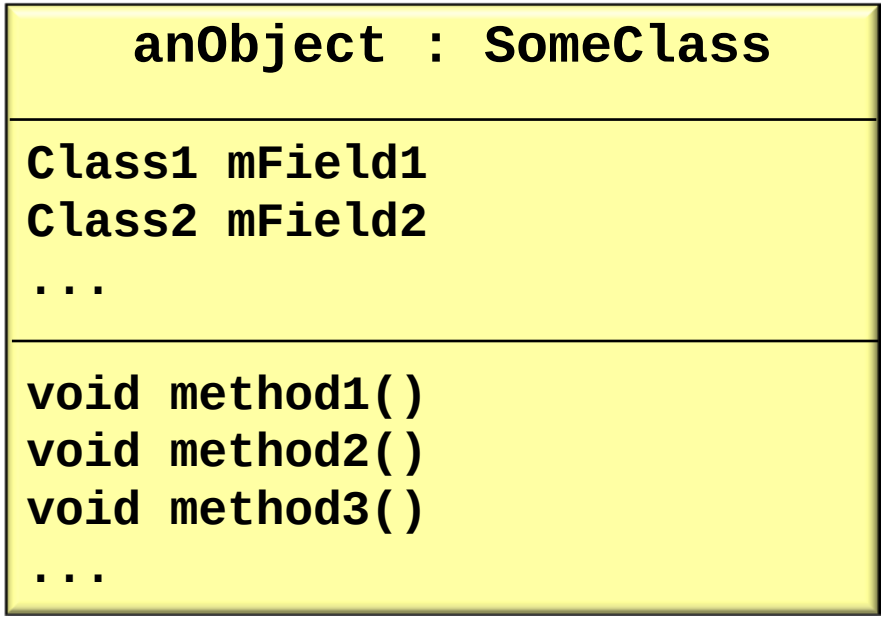
- Apps written in Java are organized in terms of *structural* elements
  - e.g., classes, interfaces, & packages



See en.wikipedia.org/wiki/Software_design_pattern

Key Object-Oriented Concepts Supported by Java

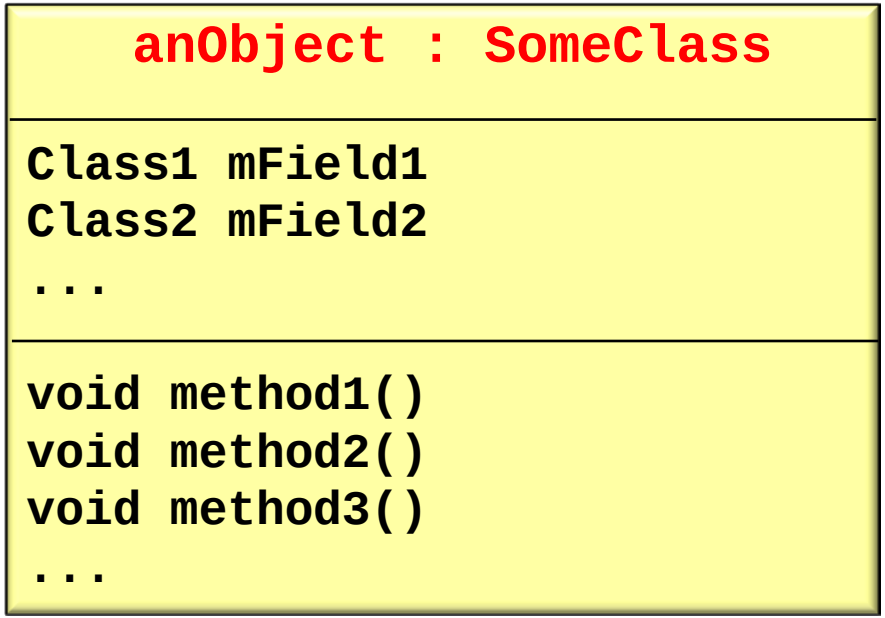
- An object is an instance of a class that performs certain operations & interacts with other objects



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

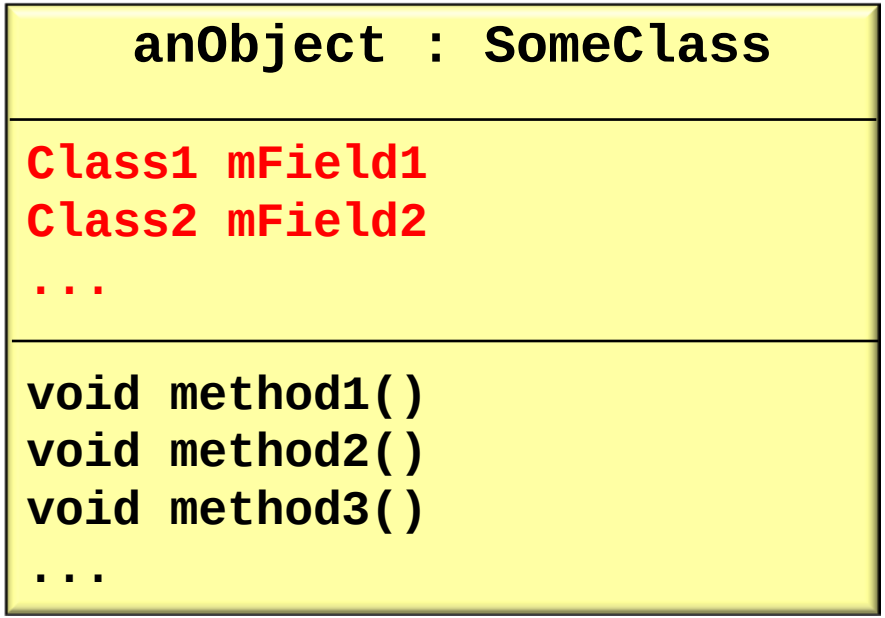
Key Object-Oriented Concepts Supported by Java

- An object is an instance of a class that performs certain operations & interacts with other objects
  - An object in Java resides in a memory location of a computer



Key Object-Oriented Concepts Supported by Java

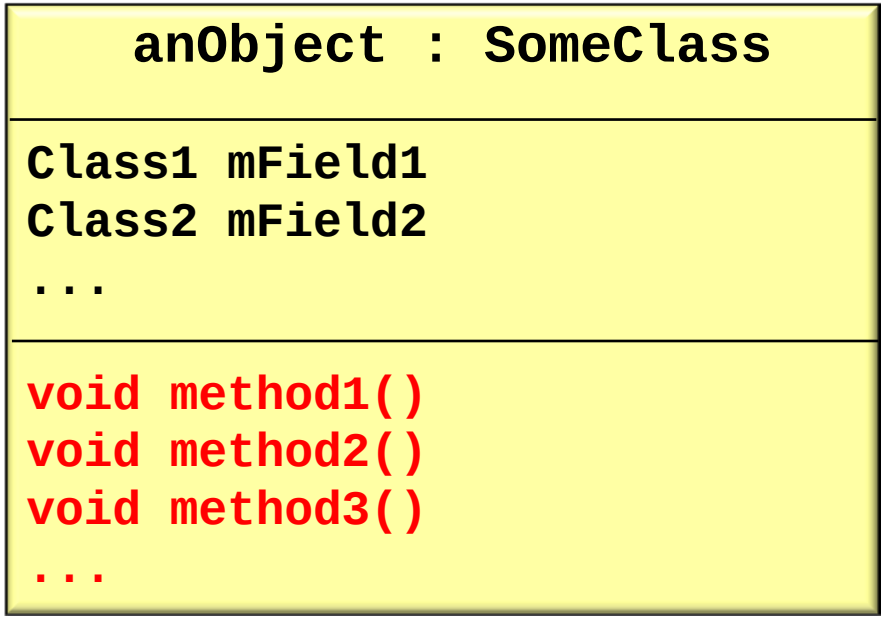
- An object is an instance of a class that performs certain operations & interacts with other objects
  - An object in Java resides in a memory location of a computer
  - It consists of
    - *State* – represented via data fields



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

Key Object-Oriented Concepts Supported by Java

- An object is an instance of a class that performs certain operations & interacts with other objects
  - An object in Java resides in a memory location of a computer
  - It consists of
    - *State* – represented via data fields
    - *Behavior* – represented via methods



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

Key Object-Oriented Concepts Supported by Java

- Objects often correspond to real-world entities

anAccount : Account
Money mCurrentBalance boolean mOverdraftProtection ...
void deposit(Money amount) void withdrawl(Money amount) Money checkCurrentBalance() ...

Key Object-Oriented Concepts Supported by Java

- Objects often correspond to real-world entities

anAccount : Account
Money mCurrentBalance boolean mOverdraftProtection ...
void deposit(Money amount) void withdrawl(Money amount) Money checkCurrentBalance() ...

Key Object-Oriented Concepts Supported by Java

- Objects often correspond to real-world entities

anAccount : Account
Money mCurrentBalance boolean mOverdraftProtection ...
void deposit(Money amount) void withdrawl(Money amount) Money checkCurrentBalance() ...

Key Object-Oriented Concepts Supported by Java

- Objects often correspond to real-world entities

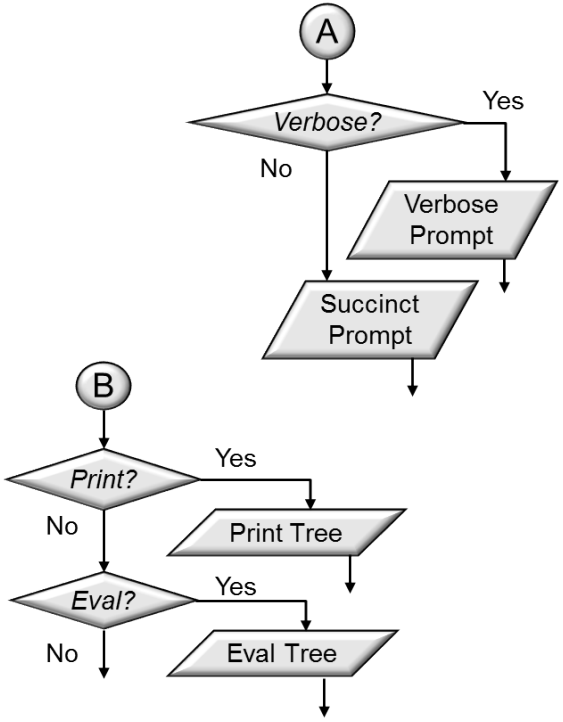
anAccount : Account
Money mCurrentBalance boolean mOverdraftProtection ...
void deposit(Money amount) void withdrawl(Money amount) Money checkCurrentBalance() ...

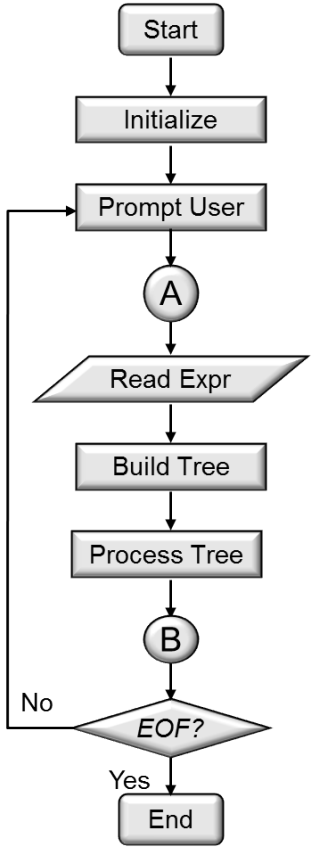
Key Object-Oriented Concepts Supported by Java

- Non-object-oriented programming languages organize apps in terms of *functional* elements

FORTRAN

THE C PROGRAMMING LANGUAGE





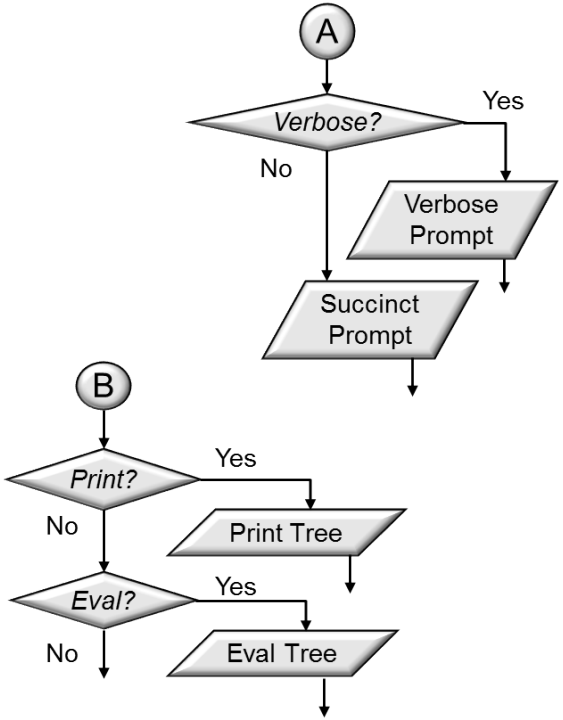
See en.wikipedia.org/wiki/Procedural_programming

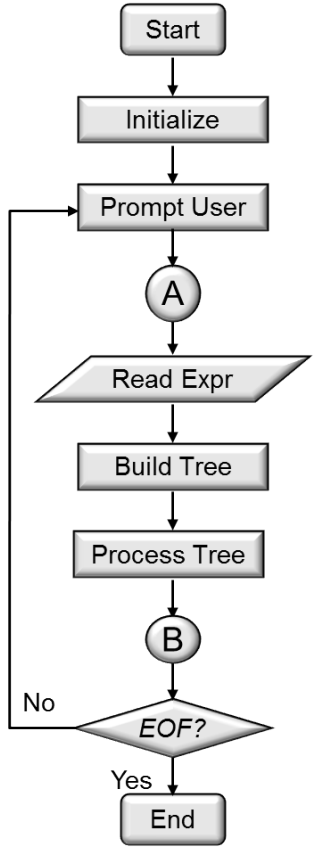
Key Object-Oriented Concepts Supported by Java

- Non-object-oriented programming languages organize apps in terms of *functional* elements
  - e.g., actions & logic

FORTRAN

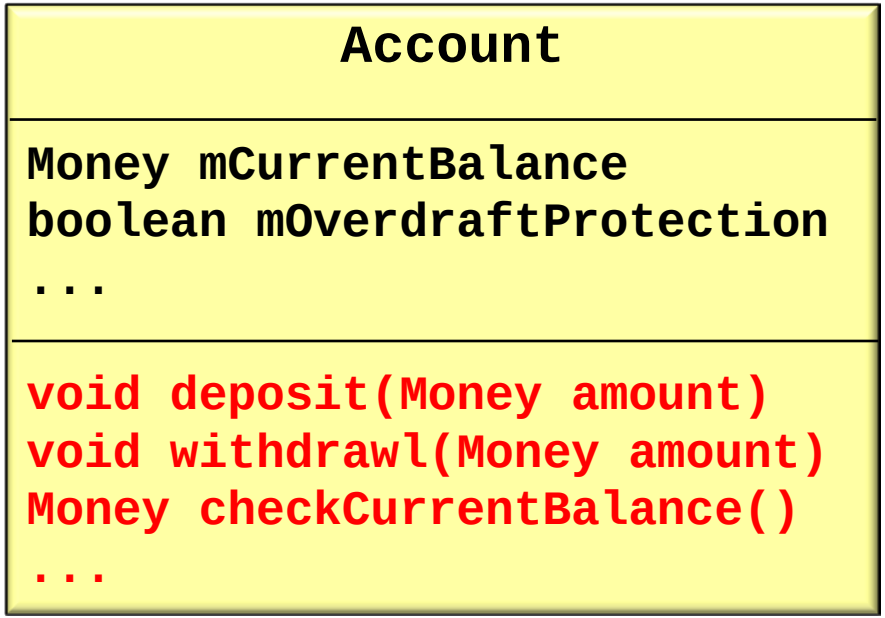
THE C PROGRAMMING LANGUAGE





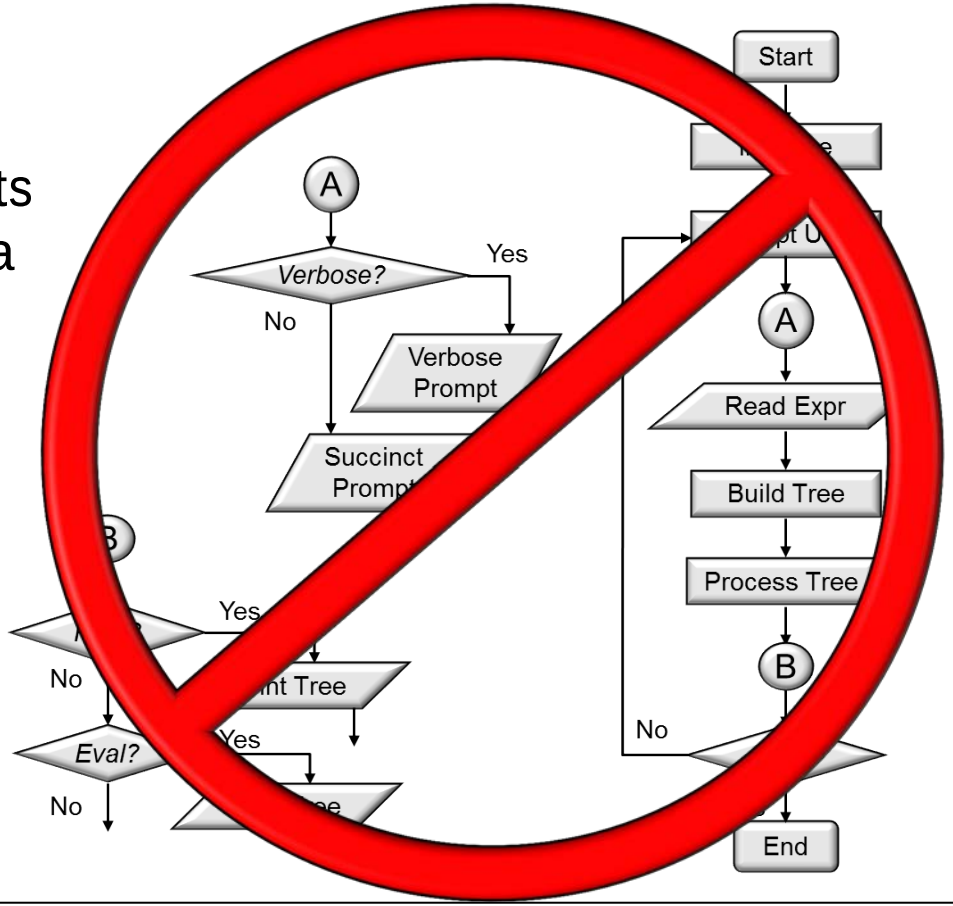
Key Object-Oriented Concepts Supported by Java

- Object-oriented Java programs also perform actions & contain logic



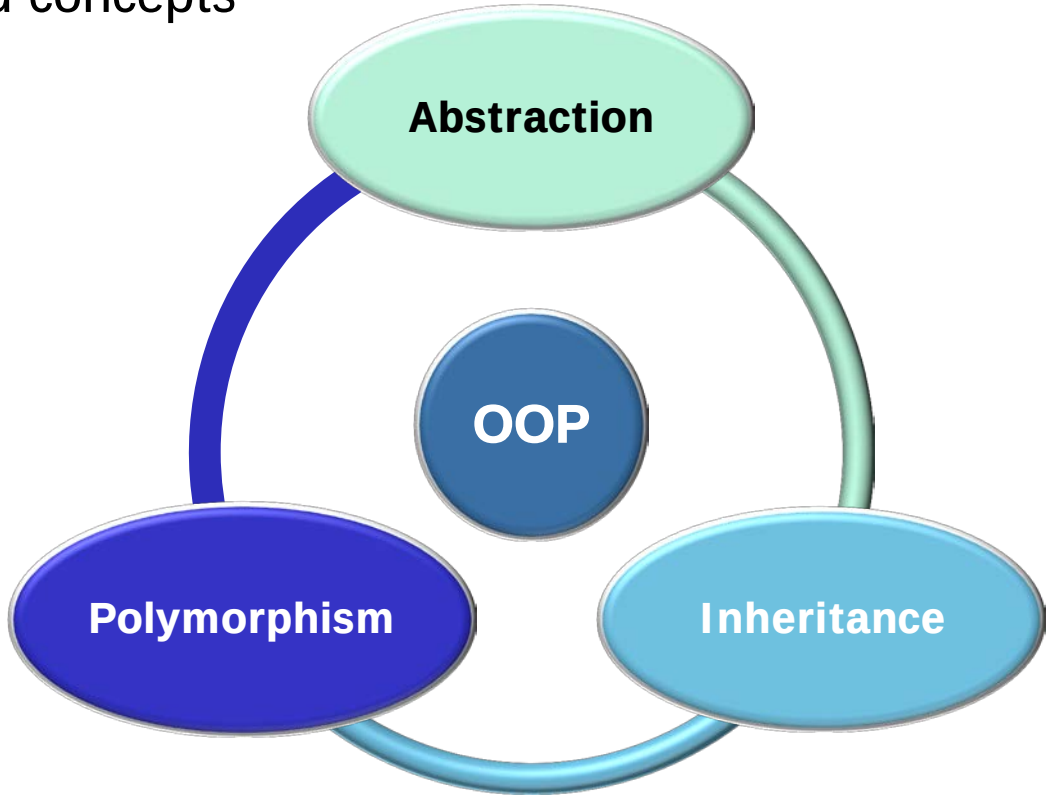
Key Object-Oriented Concepts Supported by Java

- Object-oriented Java programs also perform actions & contain logic
  - However, these functional elements don't constitute main focus in Java



Key Object-Oriented Concepts Supported by Java

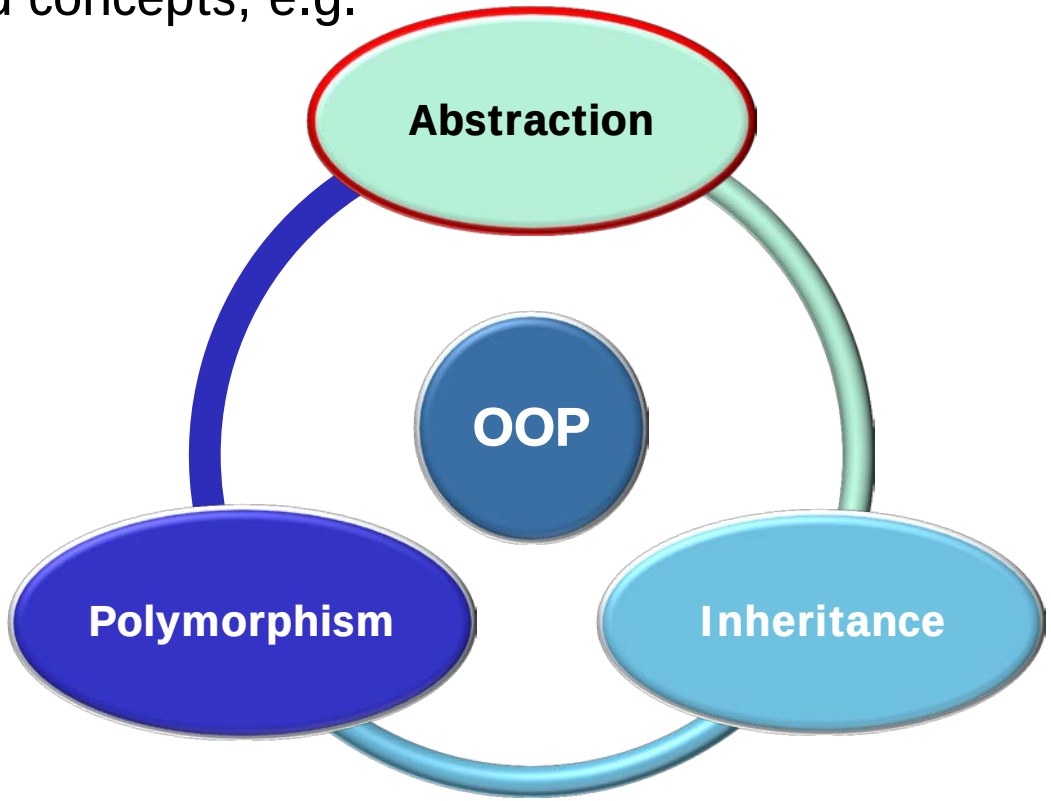
- Java supports key object-oriented concepts



See www.stroustrup.com/whatis.pdf

Key Object-Oriented Concepts Supported by Java

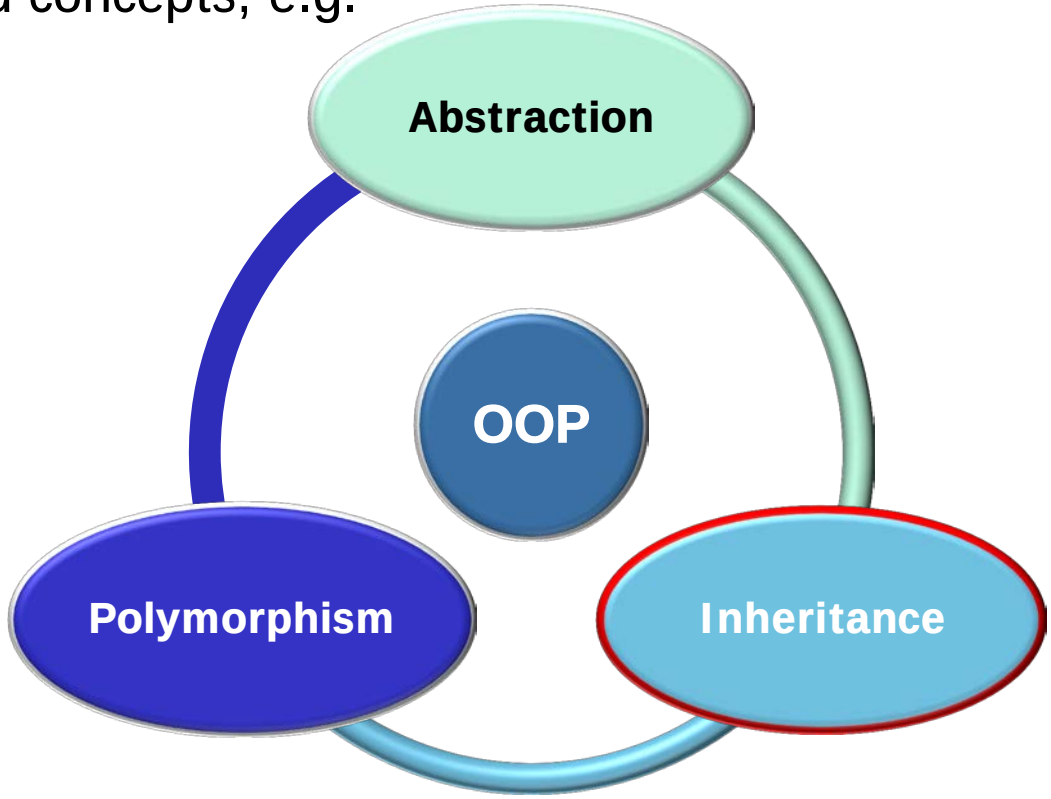
- Java supports key object-oriented concepts, e.g.
  - Data & control abstractions



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

Key Object-Oriented Concepts Supported by Java

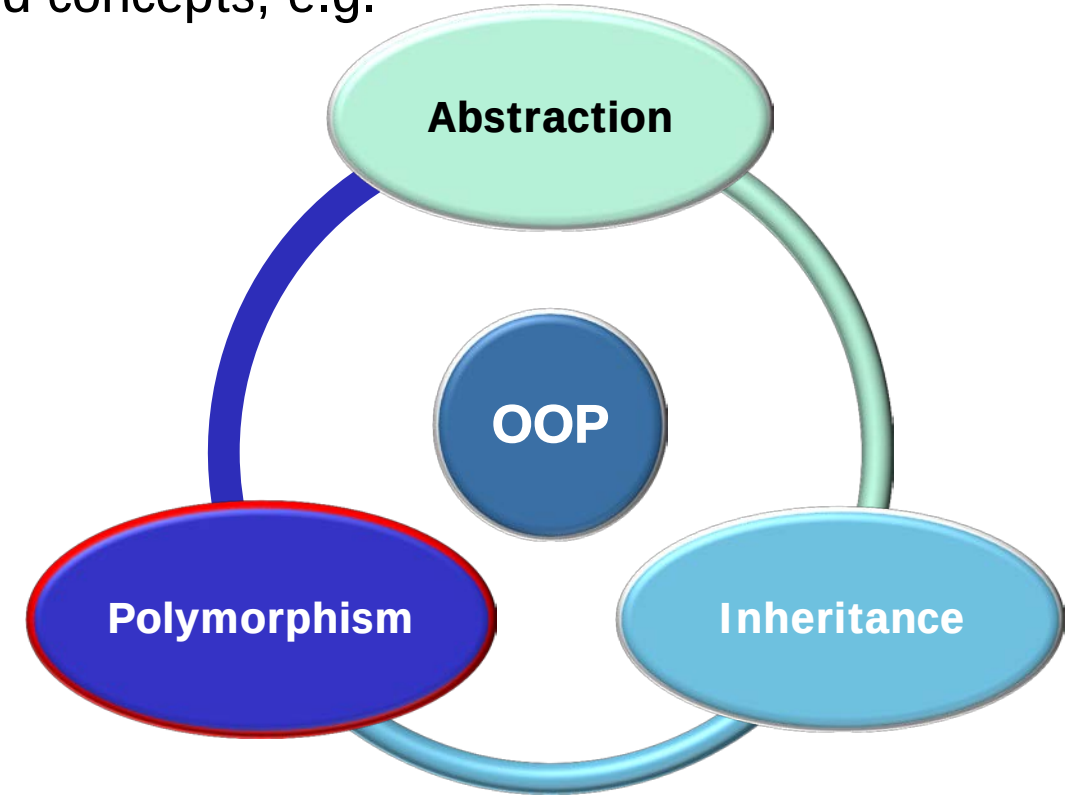
- Java supports key object-oriented concepts, e.g.
  - Data & control abstractions
  - Inheritance



See en.wikipedia.org/wiki/Inheritance_(object-oriented_programming)

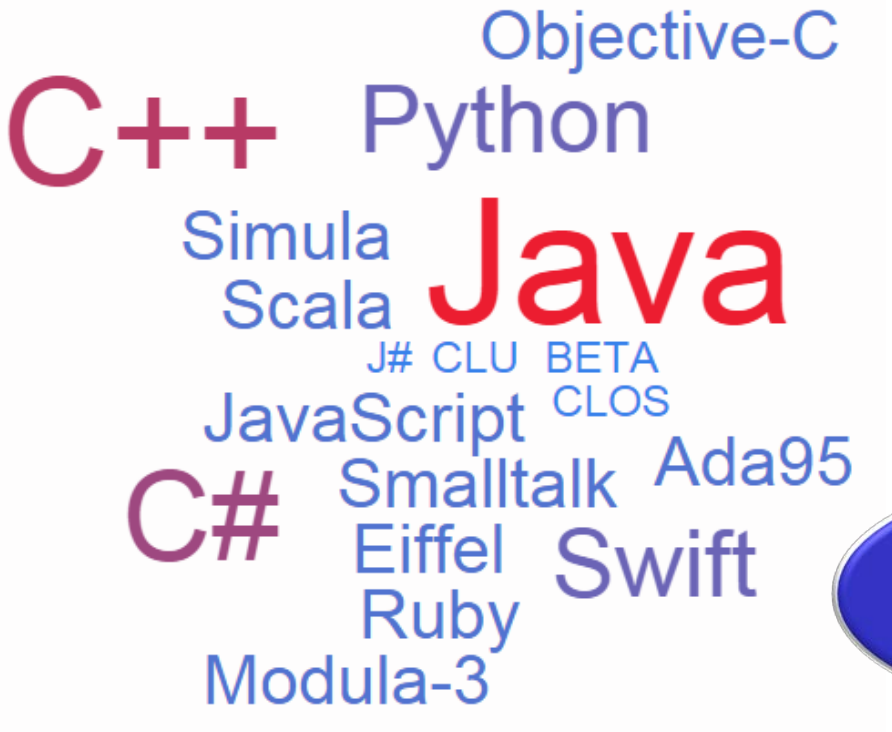
Key Object-Oriented Concepts Supported by Java

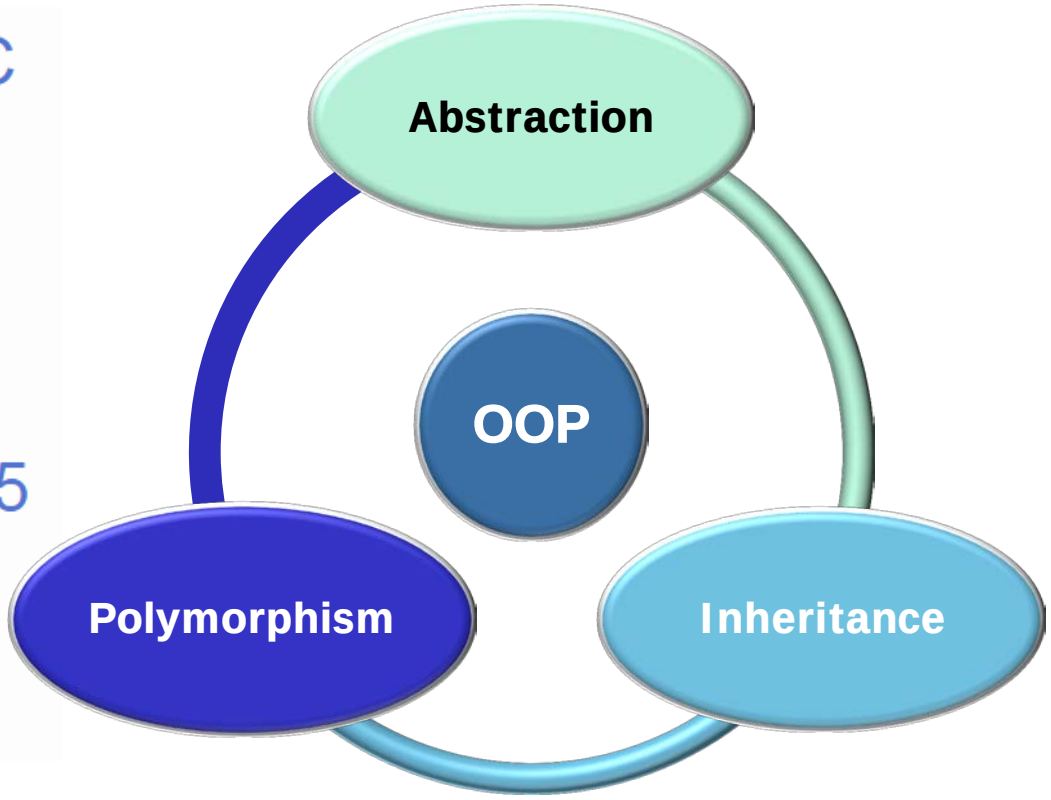
- Java supports key object-oriented concepts, e.g.
  - Data & control abstractions
  - Inheritance
  - Polymorphism



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

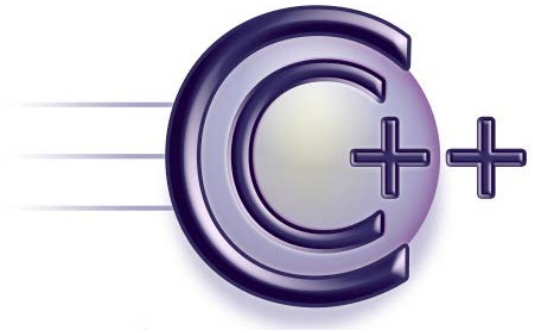
Key Object-Oriented Concepts Supported by Java

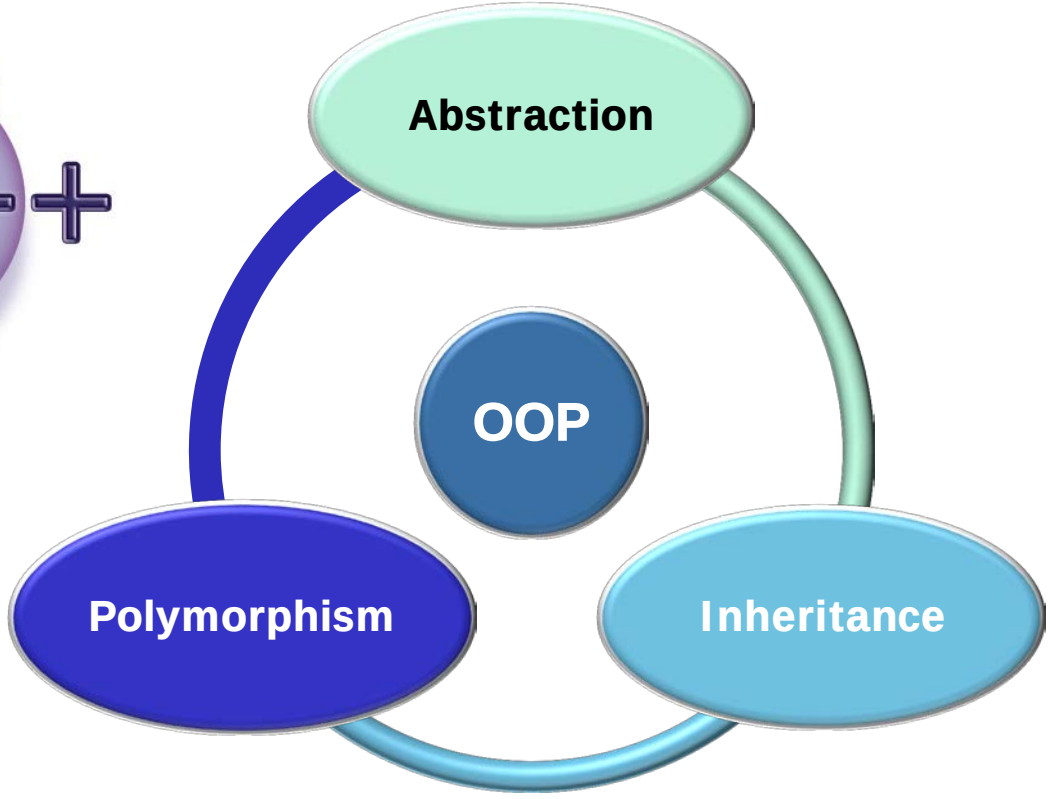




See en.wikipedia.org/wiki/List_of_object-oriented_programming_languages

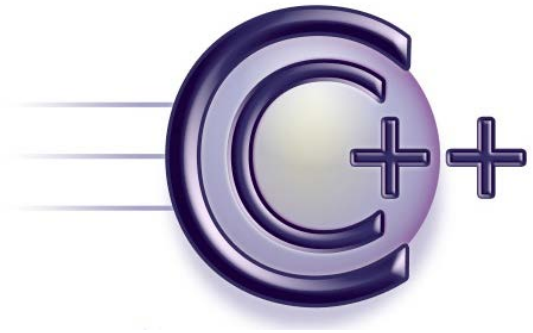
Key Object-Oriented Concepts Supported by Java

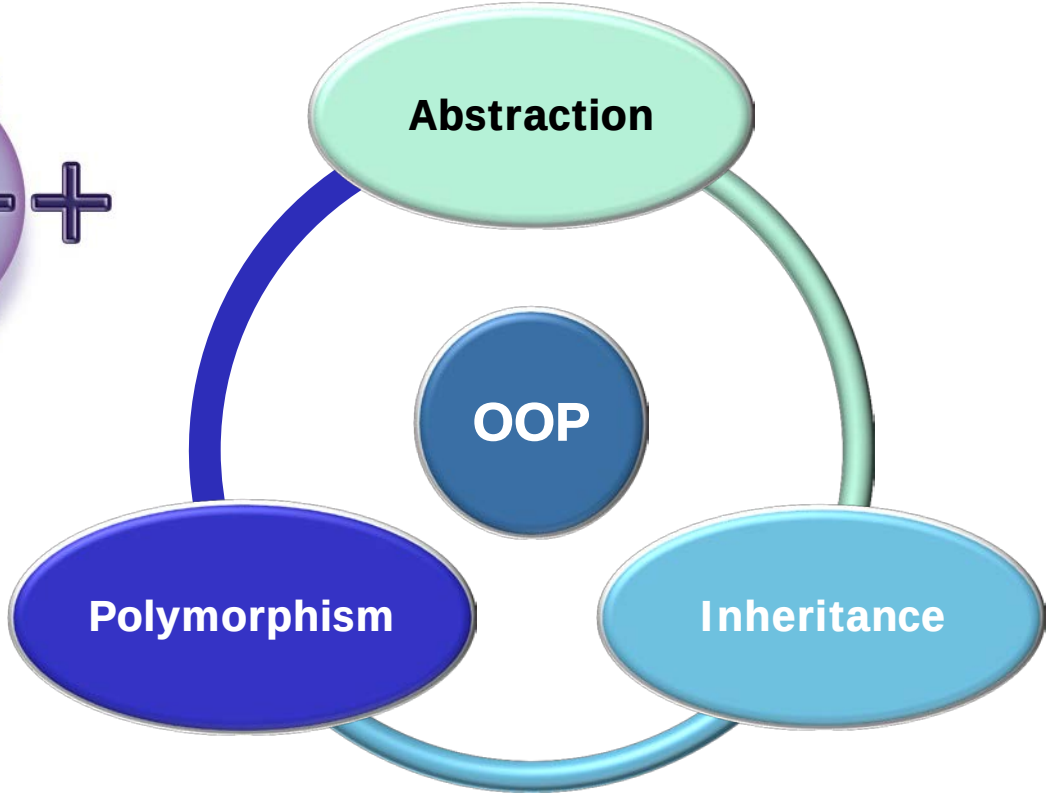




See en.wikipedia.org/wiki/List_of_object-oriented_programming_languages

Key Object-Oriented Concepts Supported by Java





Learning other object-oriented languages is much easier once you know Java

Key Object-Oriented Concepts Supported by Java

- If you already known Java you may be bored by some parts of this lesson!

Key Object-Oriented Concepts Supported by Java

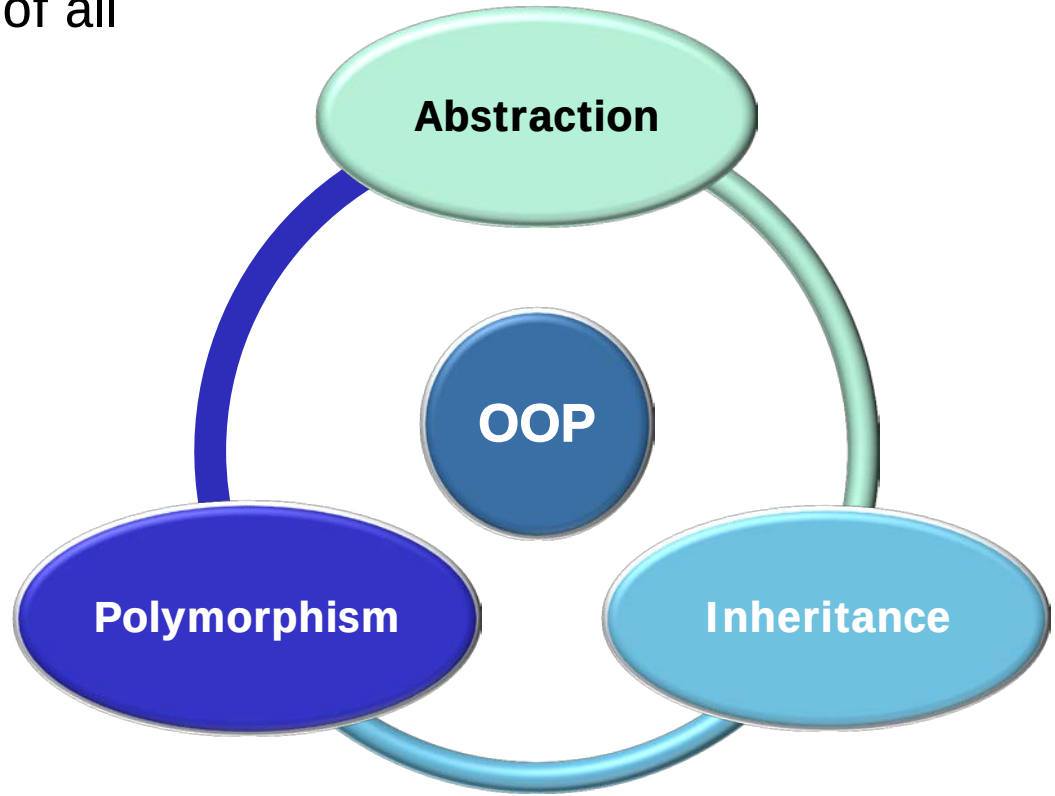
- If you already known Java you may be bored by some parts of this lesson!
  - You can move quickly through this material to prepare for the next lesson

Make sure you understand this material since other lessons depend on it..

Overview of Java's Support for Abstraction

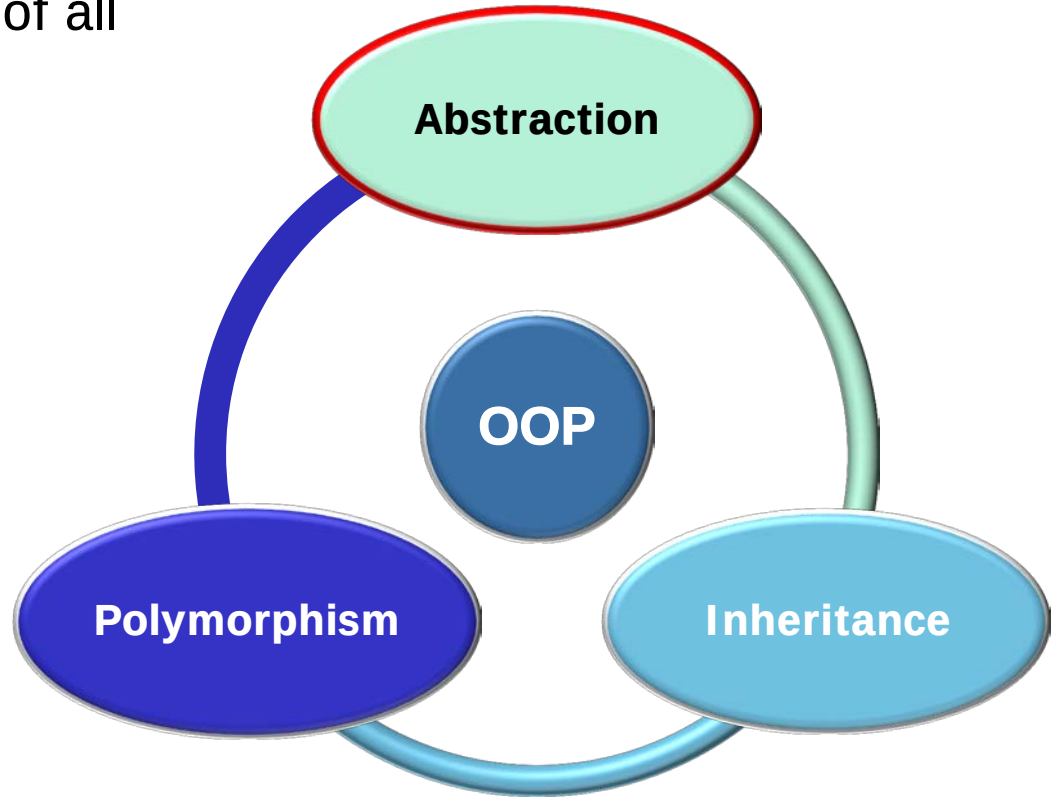
Overview of Java's Support for Abstraction

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



Overview of Java's Support for Abstraction

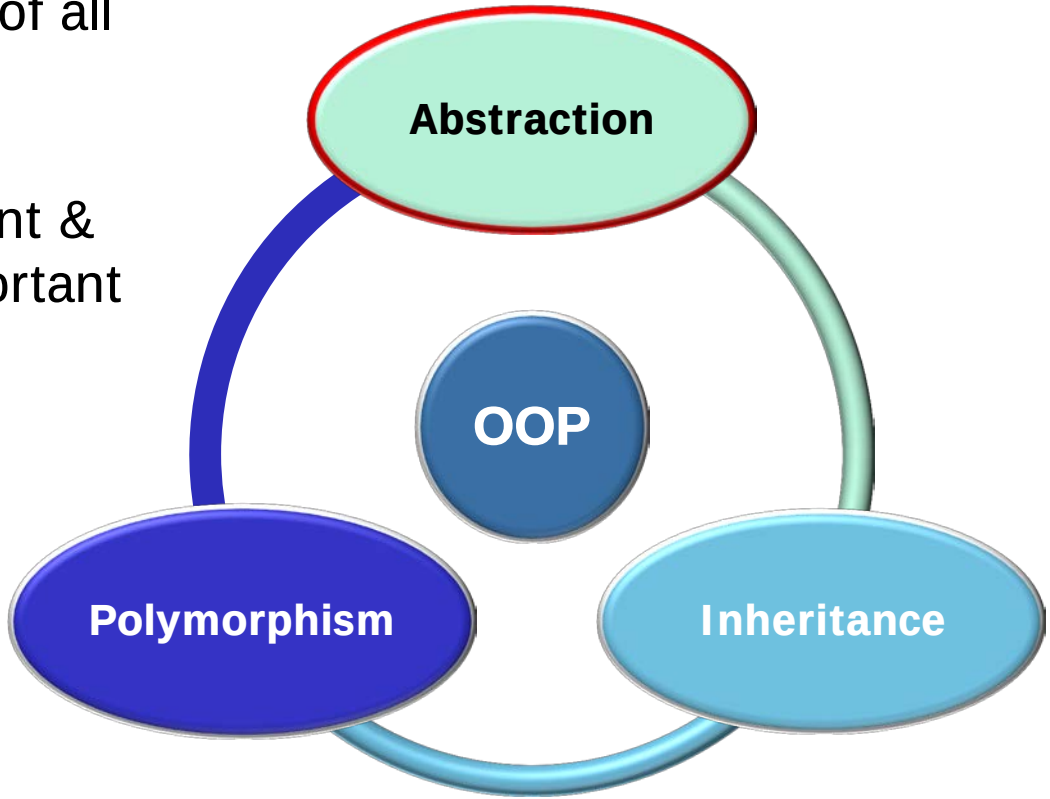
- Abstraction is an essential part of all object-oriented programming languages



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 & de-emphasizes what's unimportant at a particular level of detail



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

Overview of Java's Support for Abstraction

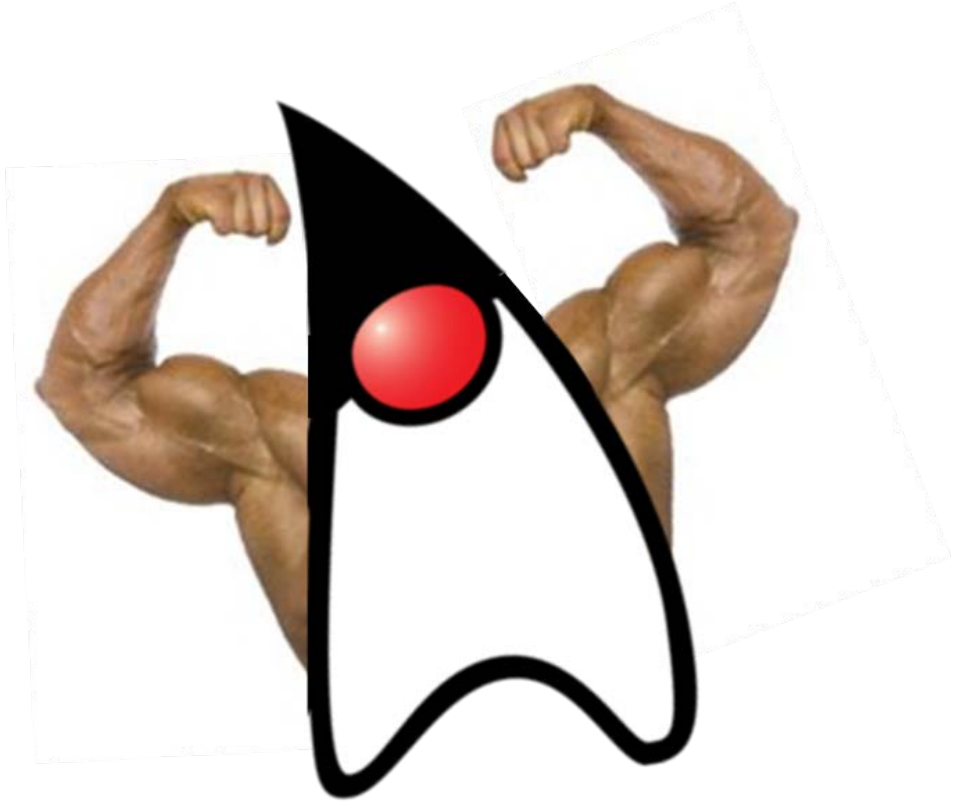
- Abstraction is an essential part of all object-oriented programming languages
  - It emphasizes what's important & de-emphasizes what's unimportant at a particular level of detail

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 at a particular level of detail

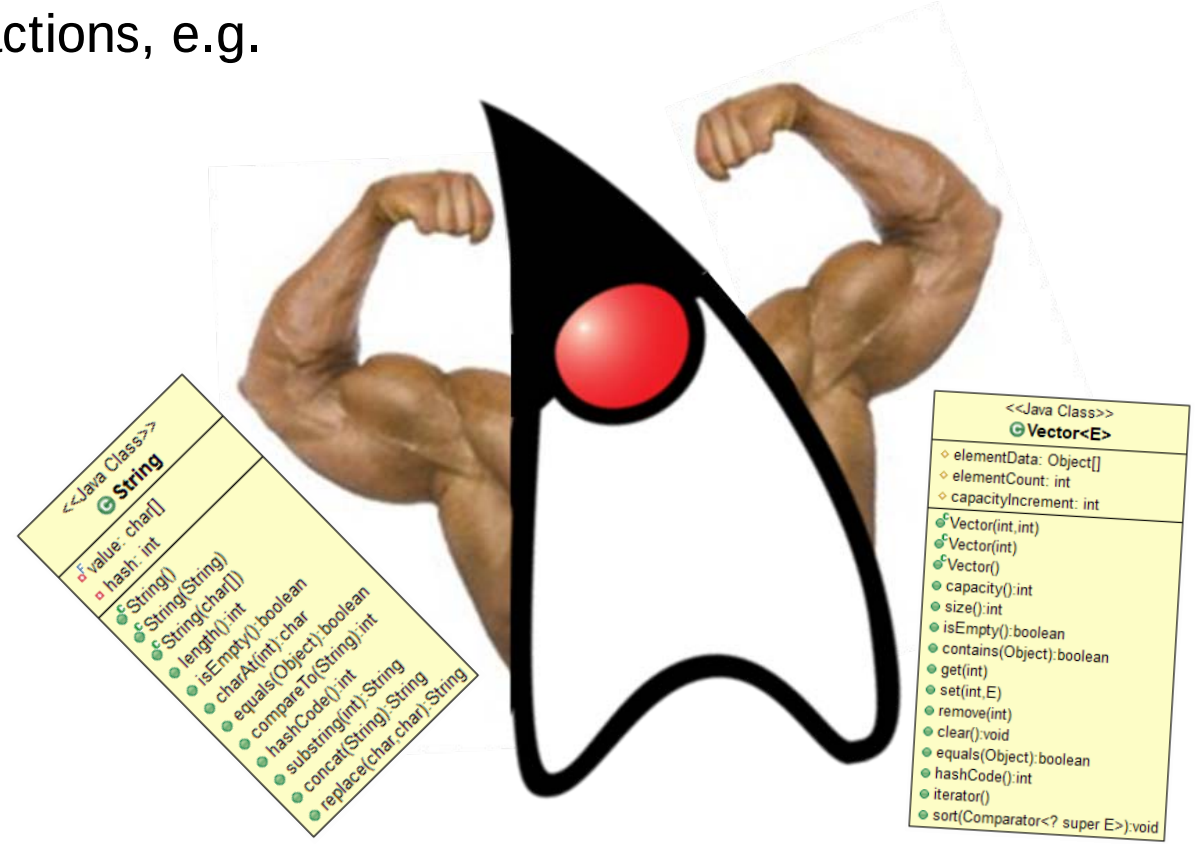
Overview of Java's Support for Abstraction

- Java supports many abstractions



Overview of Java's Support for Abstraction

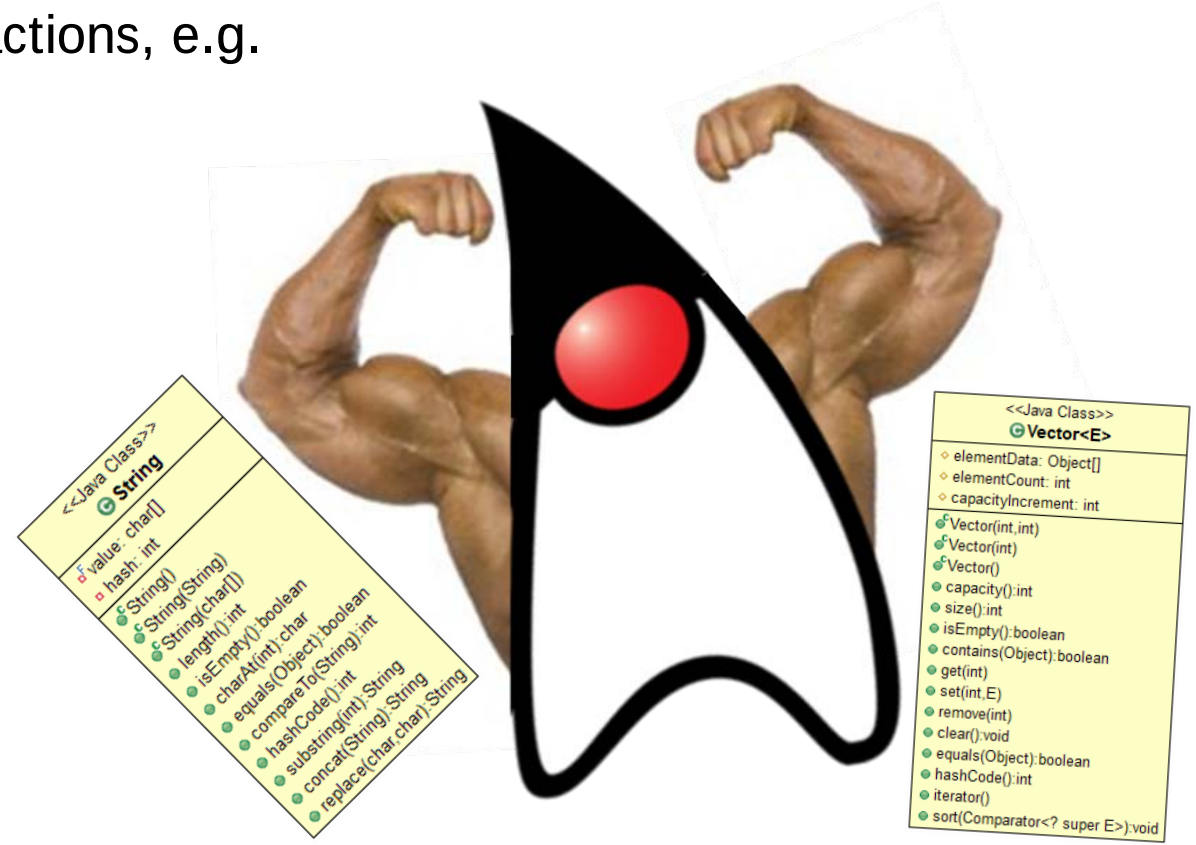
- Java supports many abstractions, e.g.
  - Data abstractions



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

Overview of Java's Support for Abstraction

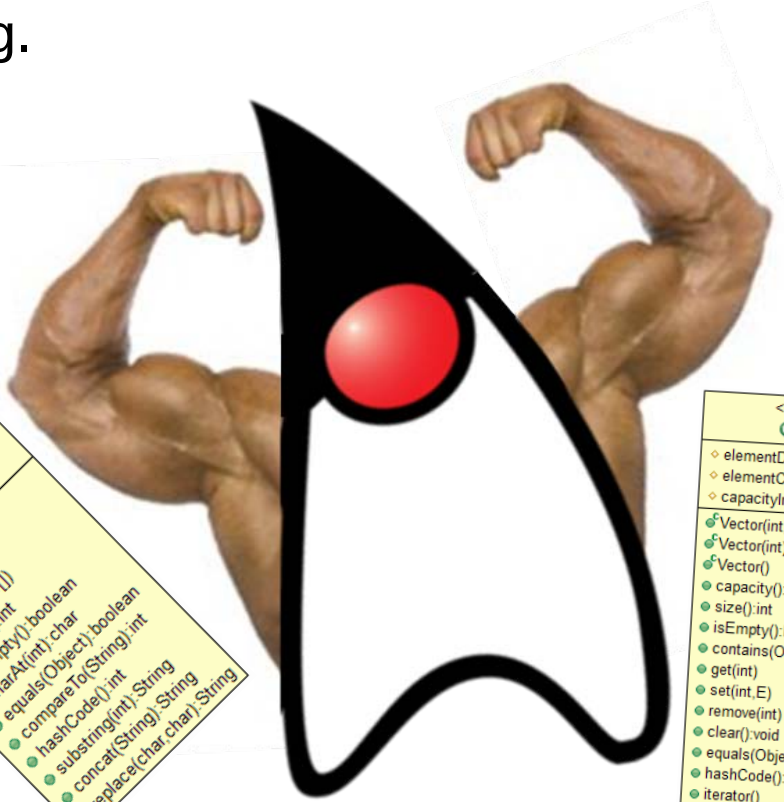
- Java supports many abstractions, e.g.
  - Data abstractions

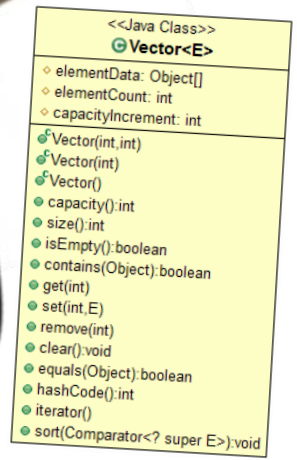


See en.wikipedia.org/wiki/Application_programming_interface

Overview of Java's Support for Abstraction

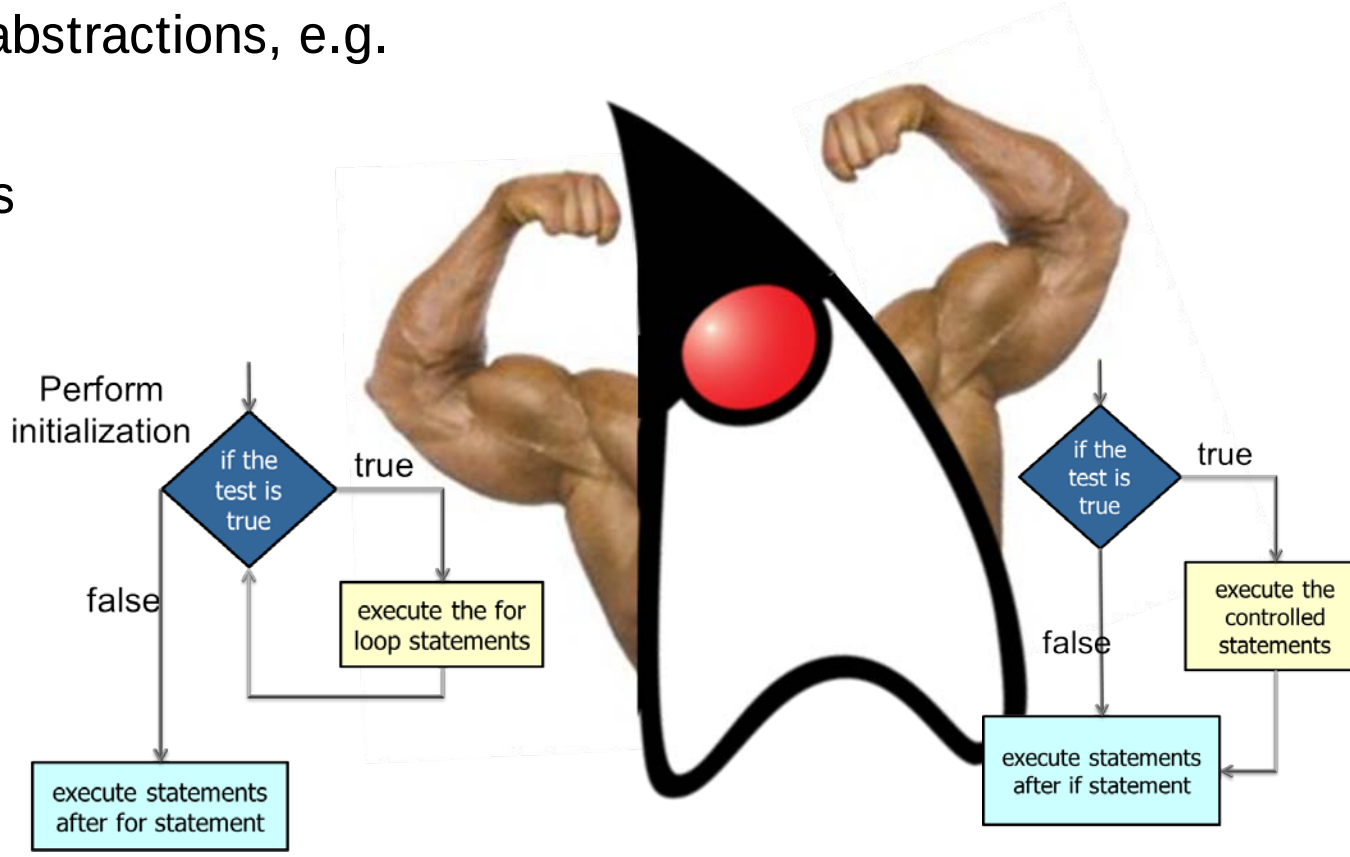
- Java supports many abstractions, e.g.
  - Data abstractions





Overview of Java's Support for Abstraction

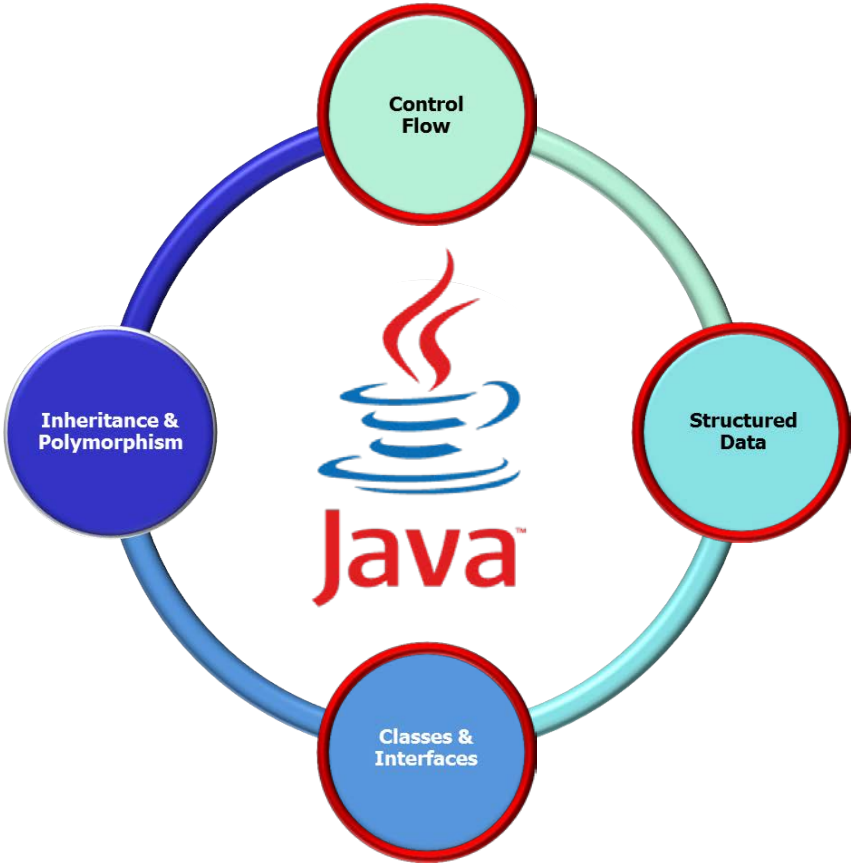
- Java supports many abstractions, e.g.
  - Data abstractions
  - Control abstractions



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

Overview of Java's Support for Abstraction

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

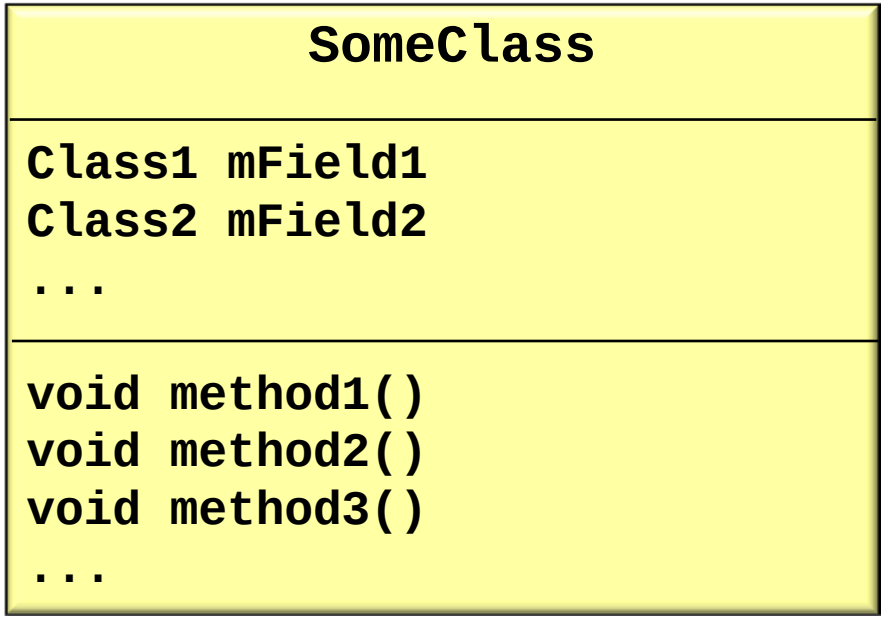


We'll now summarize various data & control abstractions supported by Java

Overview of Java's Support for Data Abstractions (Part 1)

Overview of Java's Support for Data Abstractions

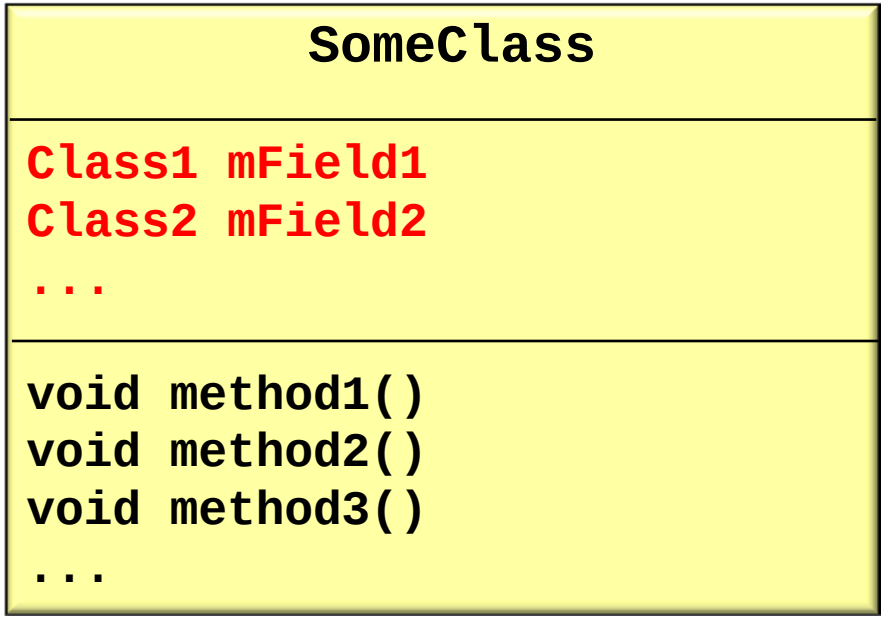
- Java supports data abstraction via Abstract Data Types (ADTs)



See en.wikipedia.org/wiki/Abstract_data_type

Overview of Java's Support for Data Abstractions

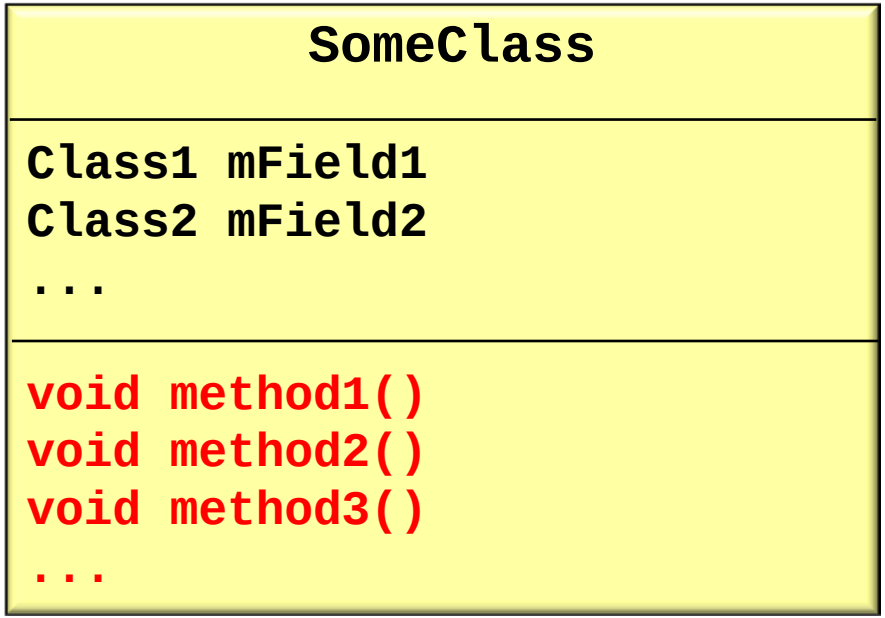
- Java supports data abstraction via Abstract Data Types (ADTs), which define
  - A set of data values



See en.wikipedia.org/wiki/Abstract_data_type

Overview of Java's Support for Data Abstractions

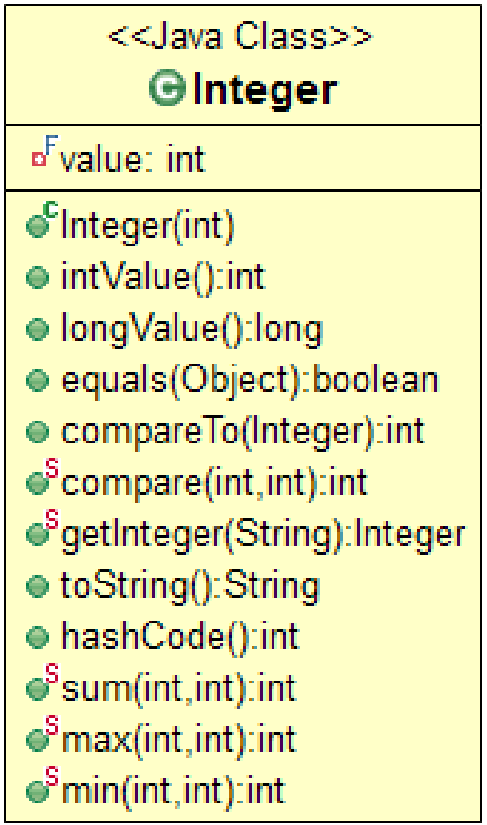
- Java supports data abstraction via Abstract Data Types (ADTs), which define
  - A set of data values
  - A set of operations on these values



See en.wikipedia.org/wiki/Abstract_data_type

Overview of Java's Support for Data Abstractions

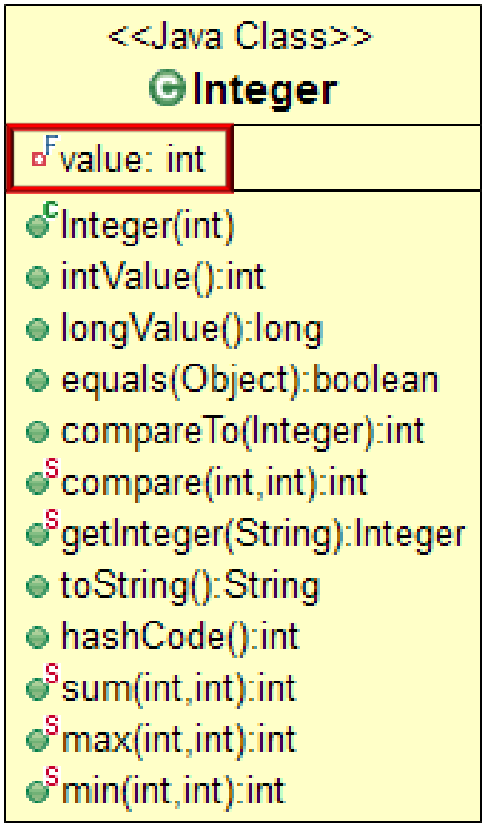
- The Java Integer class is an example of an ADT



See developer.android.com/reference/java/lang/Integer.html

Overview of Java's Support for Data Abstractions

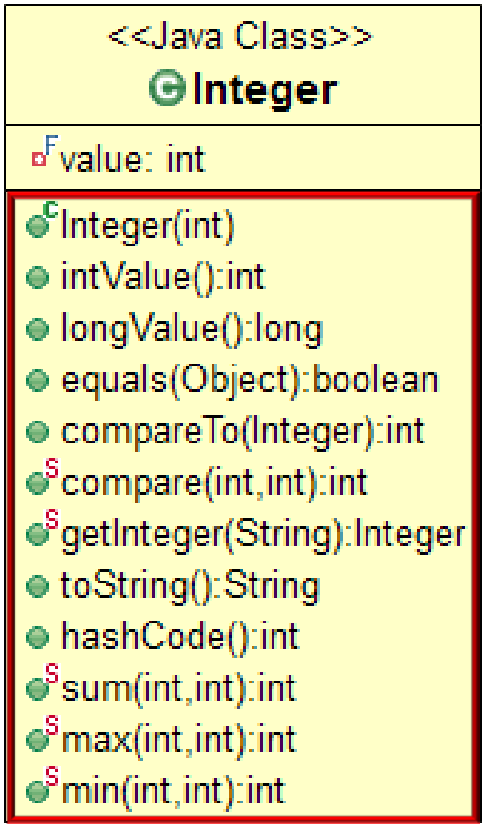
- The Java Integer class is an example of an ADT
  - It contains a value



See developer.android.com/reference/java/lang/Integer.html

Overview of Java's Support for Data Abstractions

- The Java Integer class is an example of an ADT
  - It contains a value
  - It contains operations on the value



See developer.android.com/reference/java/lang/Integer.html

Overview of Java's Support for Data Abstractions

- At the heart of data abstraction is encapsulation

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

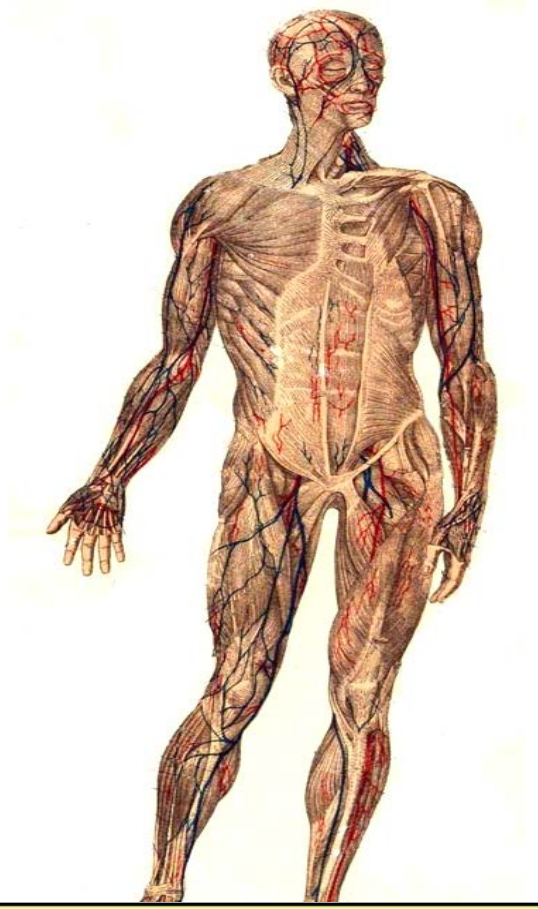
Overview of Java's Support for Data Abstractions

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

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

Overview of Java's Support for Data Abstractions

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



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

Overview of Java's Support for Data Abstractions

- Java classes provide a blueprint for creating objects

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

Overview of Java's Support for Data Abstractions

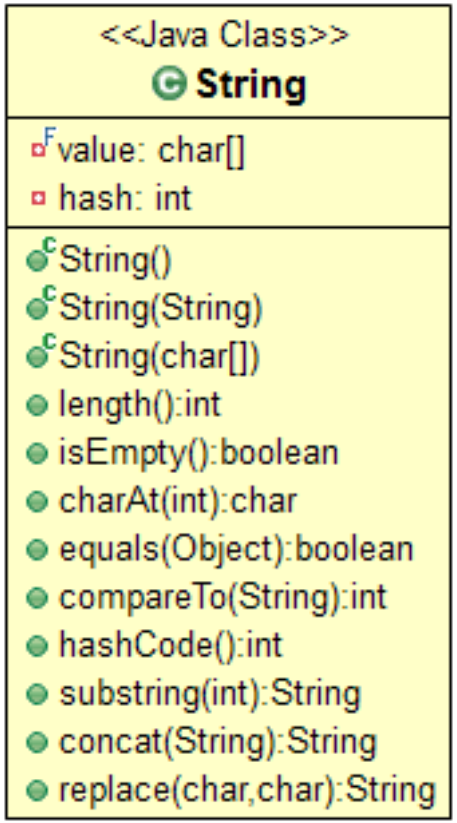
- Java classes provide a blueprint for creating objects, which provides
  - *Fields*
    - Used to store the state of an object

Overview of Java's Support for Data Abstractions

- Java classes provide a blueprint for creating objects, which provides
  - *Fields*
  - *Methods*
    - Used to implement the behaviors of an object

Overview of Java's Support for Data Abstractions

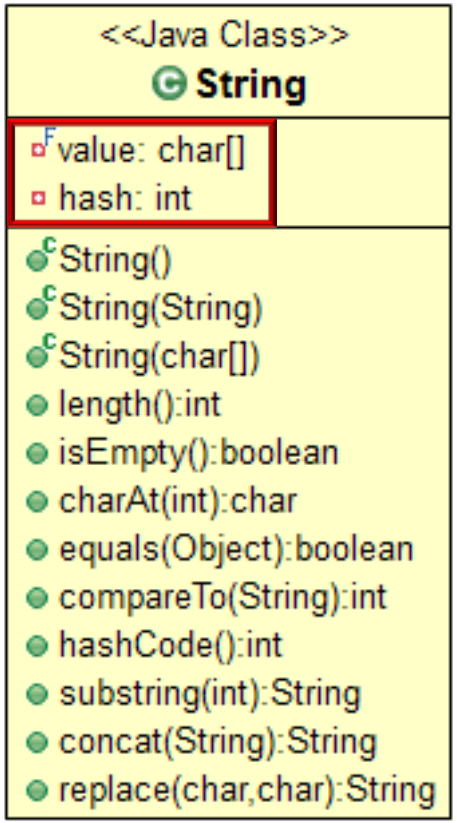
- The Java String class contains



See docs.oracle.com/javase/7/docs/api/java/lang/String.html

Overview of Java's Support for Data Abstractions

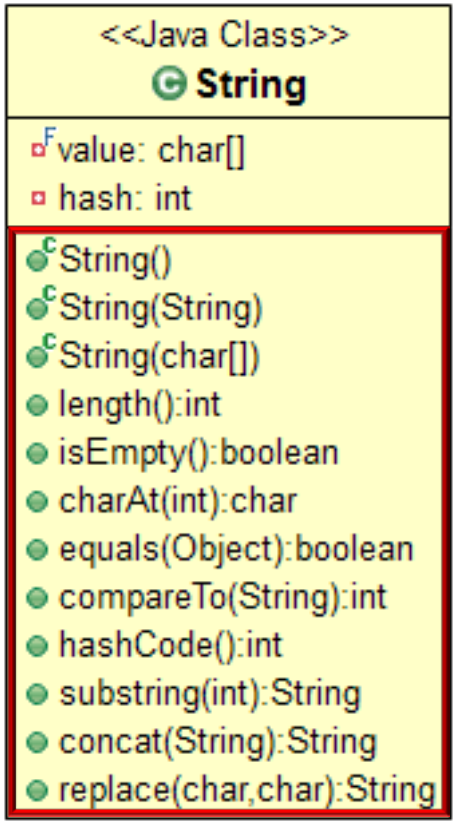
- The Java String class contains
  - Fields
    - e.g., store a sequence of characters, length of this sequence, etc.



See docs.oracle.com/javase/7/docs/api/java/lang/String.html

Overview of Java's Support for Data Abstractions

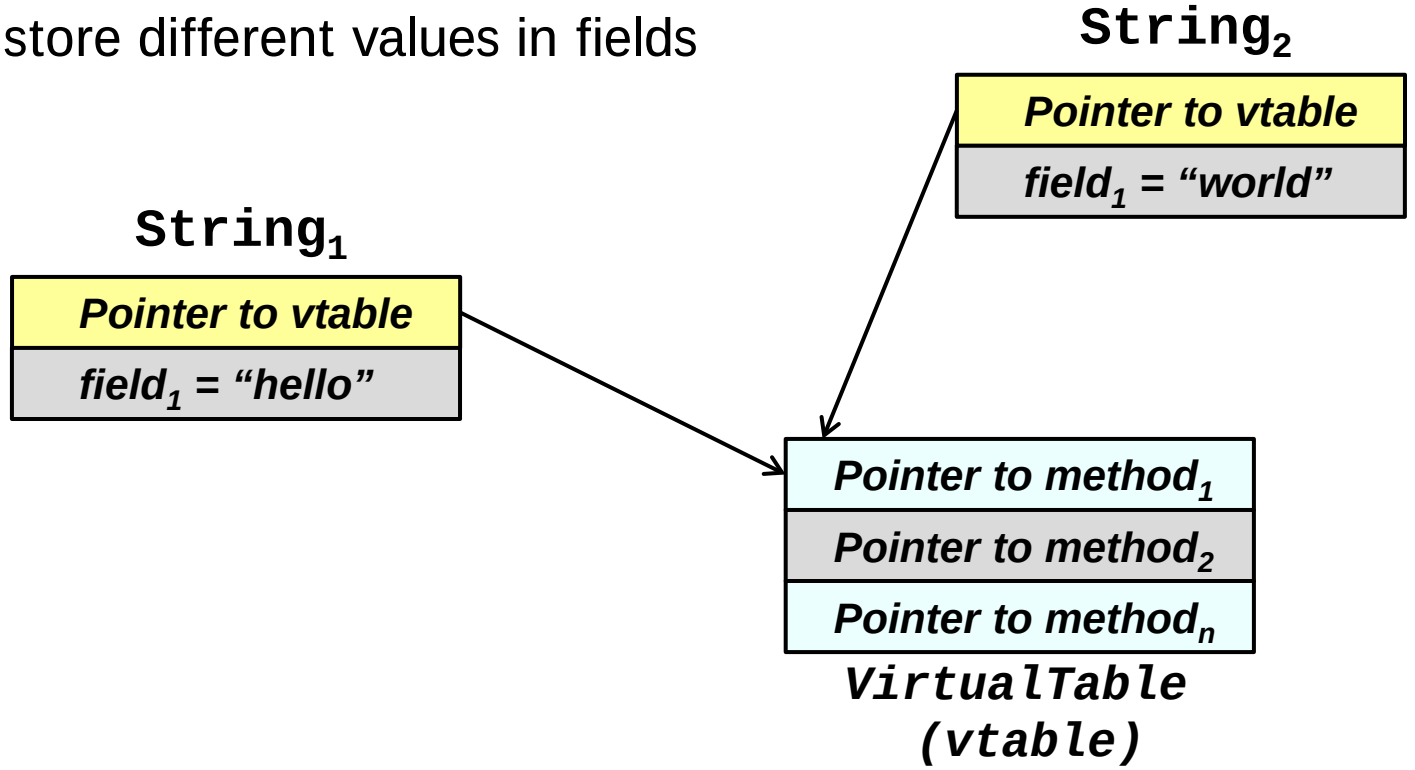
- The Java String class contains
  - Fields
  - Methods
    - e.g., examine individual characters of the sequence, compare strings, search strings, extract substrings, create copies of a string, etc.



See docs.oracle.com/javase/7/docs/api/java/lang/String.html

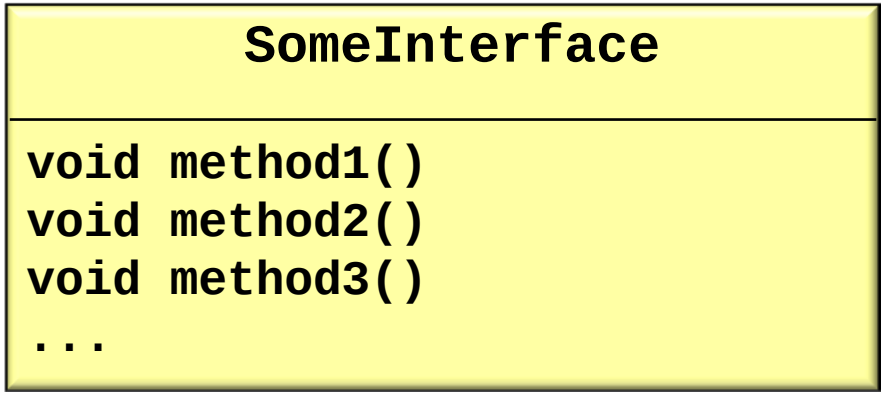
Overview of Java's Support for Data Abstractions

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



Overview of Java's Support for Data Abstractions

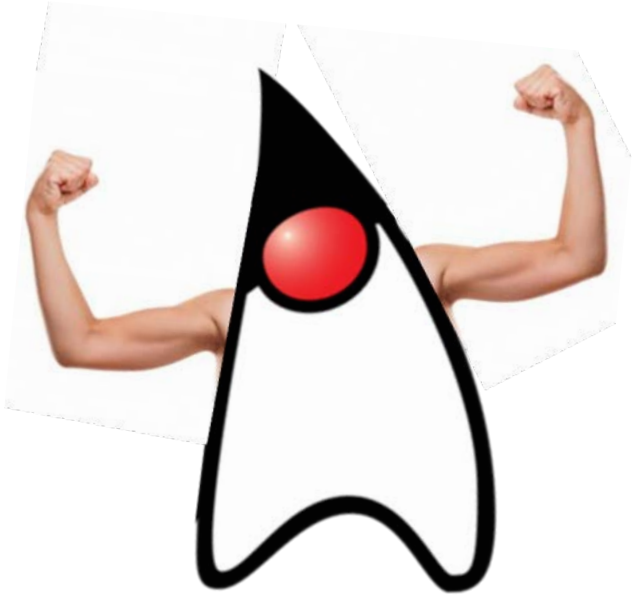
- Java interfaces define a contract specifying methods that classes implementing the interface provide



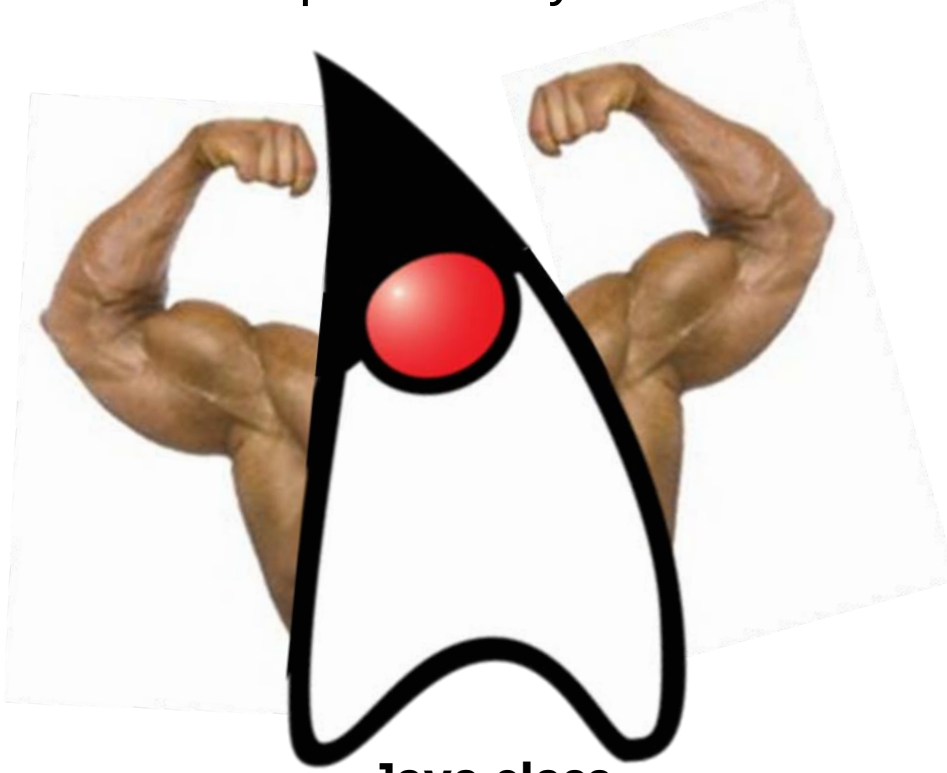
See docs.oracle.com/javase/tutorial/java/concepts/interface.html

Overview of Java's Support for Data Abstractions

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



Java interface

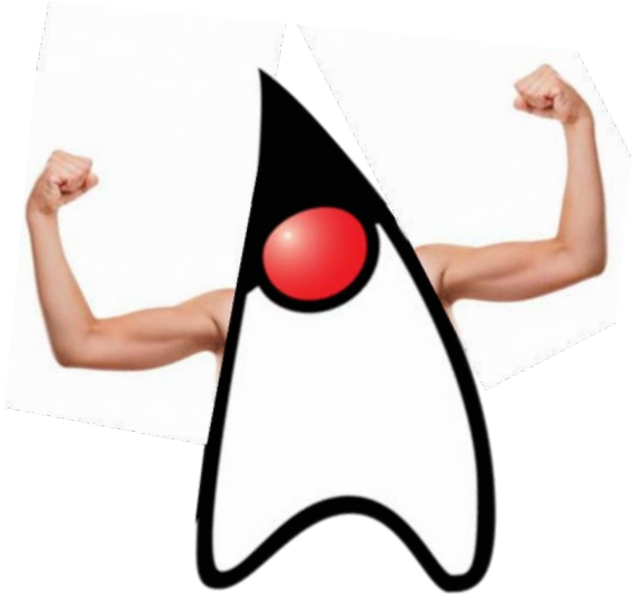


Java class

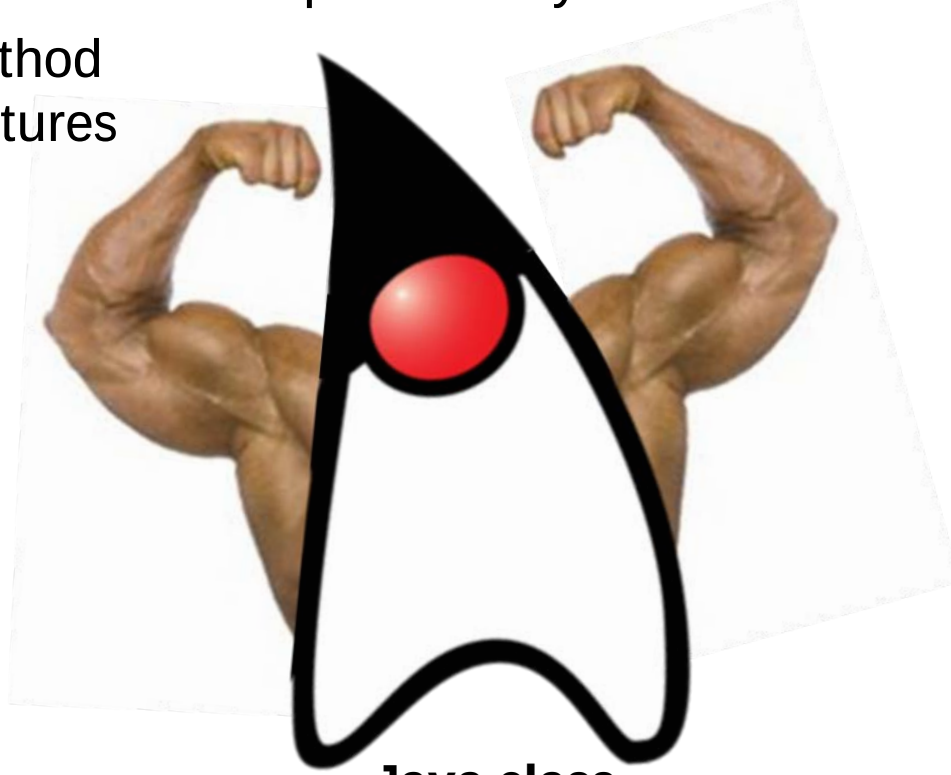
See www.tutorialspoint.com/java/java_interfaces.htm

Overview of Java's Support for Data Abstractions

- A Java interfaces provides a subset of the features provided by a Java class
  - e.g., an interface cannot contain method implementations, only method signatures



Java interface

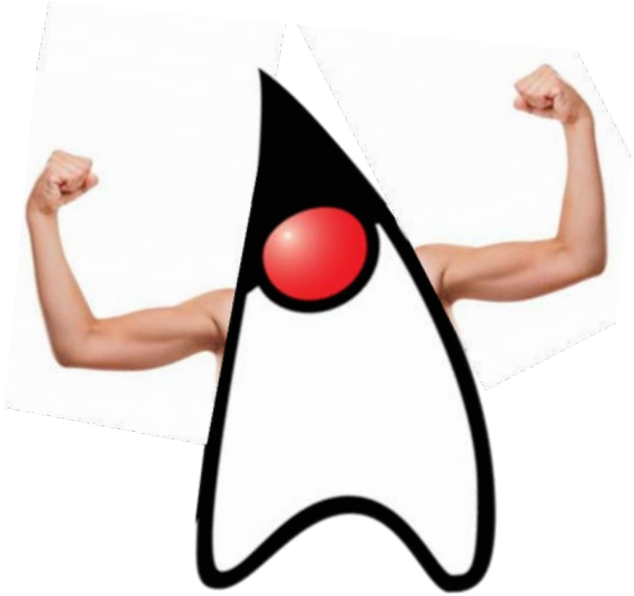


Java class

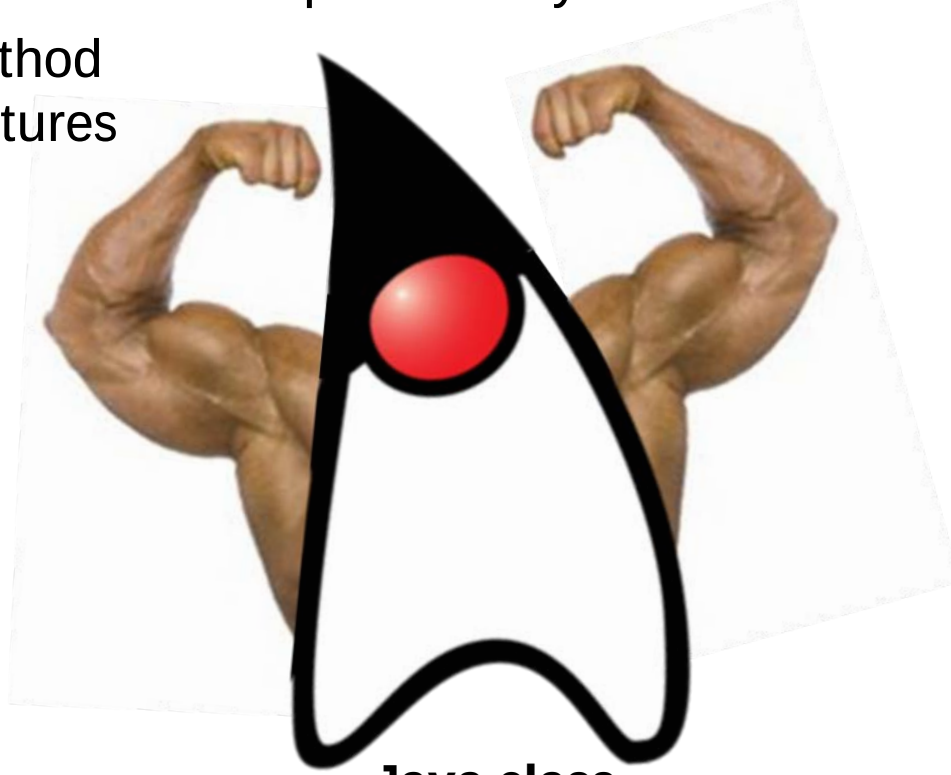
See en.wikipedia.org/wiki/Type_signature#Java_2

Overview of Java's Support for Data Abstractions

- A Java interfaces provides a subset of the features provided by a Java class
  - e.g., an interface cannot contain method implementations, only method signatures



Java interface

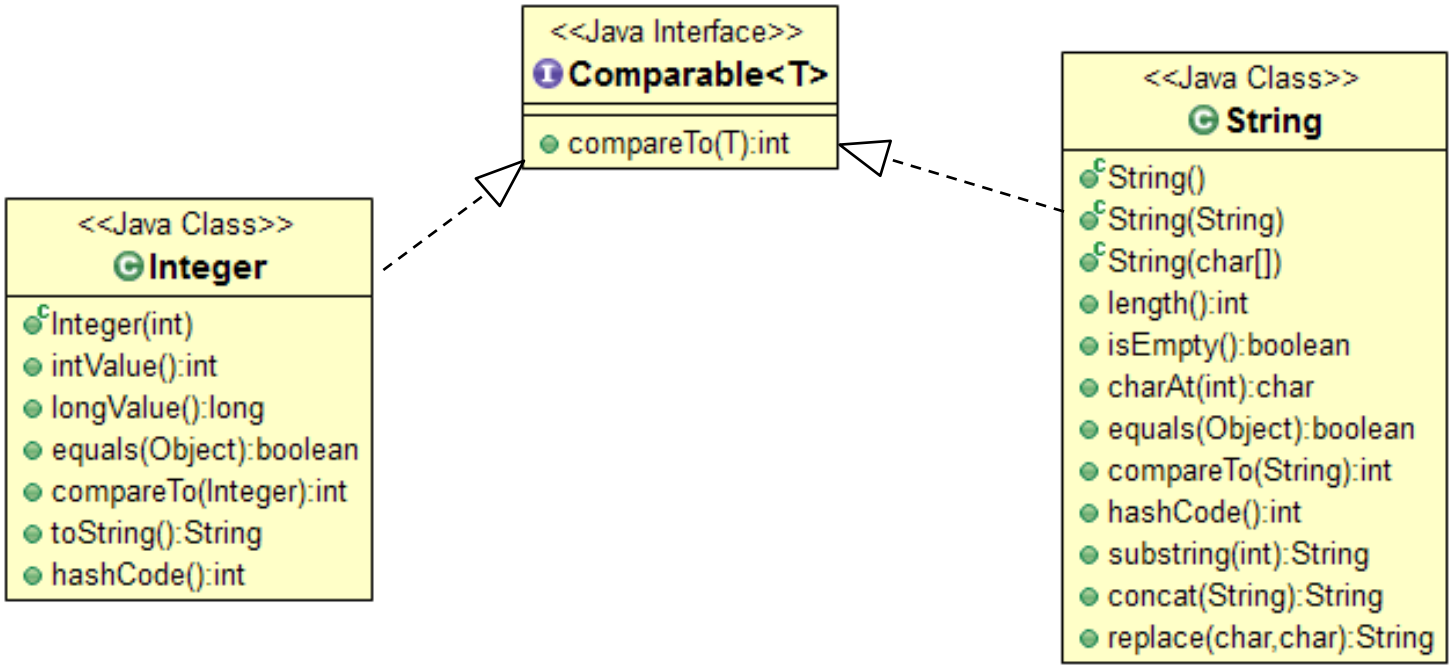


Java class

Java 8 supports "default methods" in interfaces

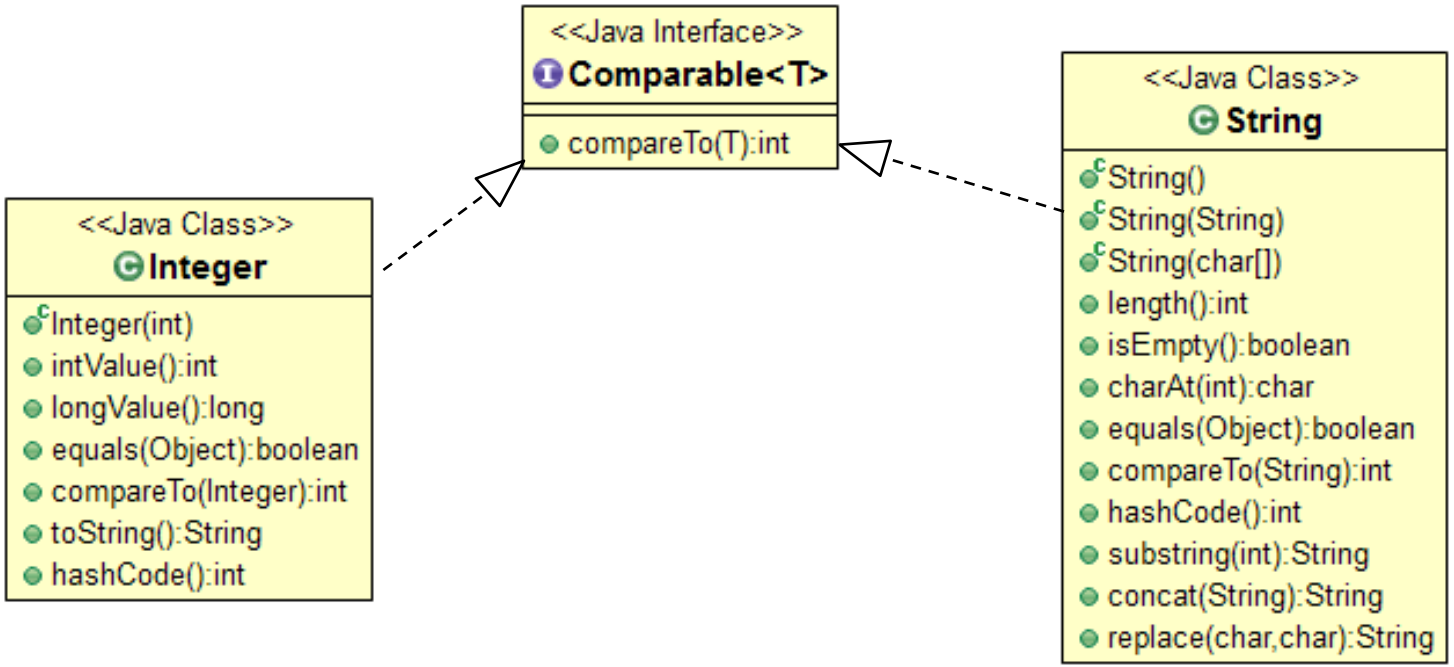
Overview of Java's Support for Data Abstractions

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



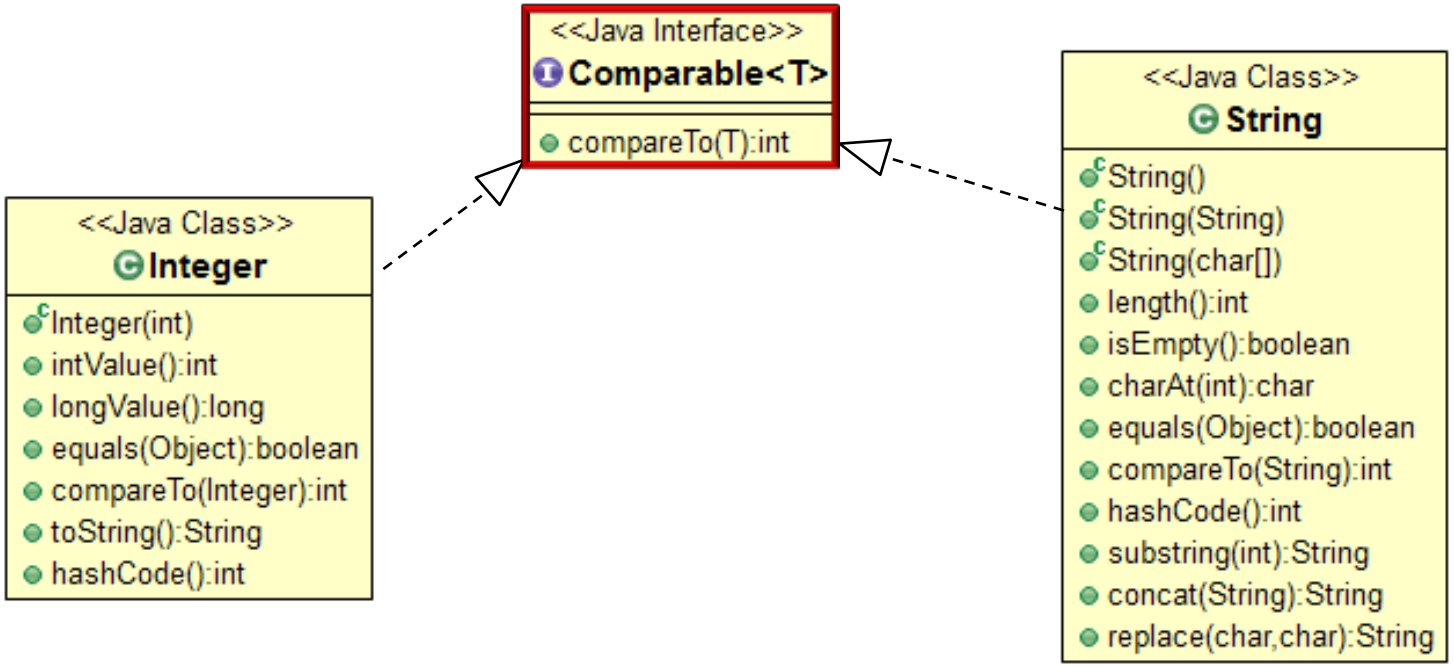
Overview of Java's Support for Data Abstractions

- 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 Data Abstractions

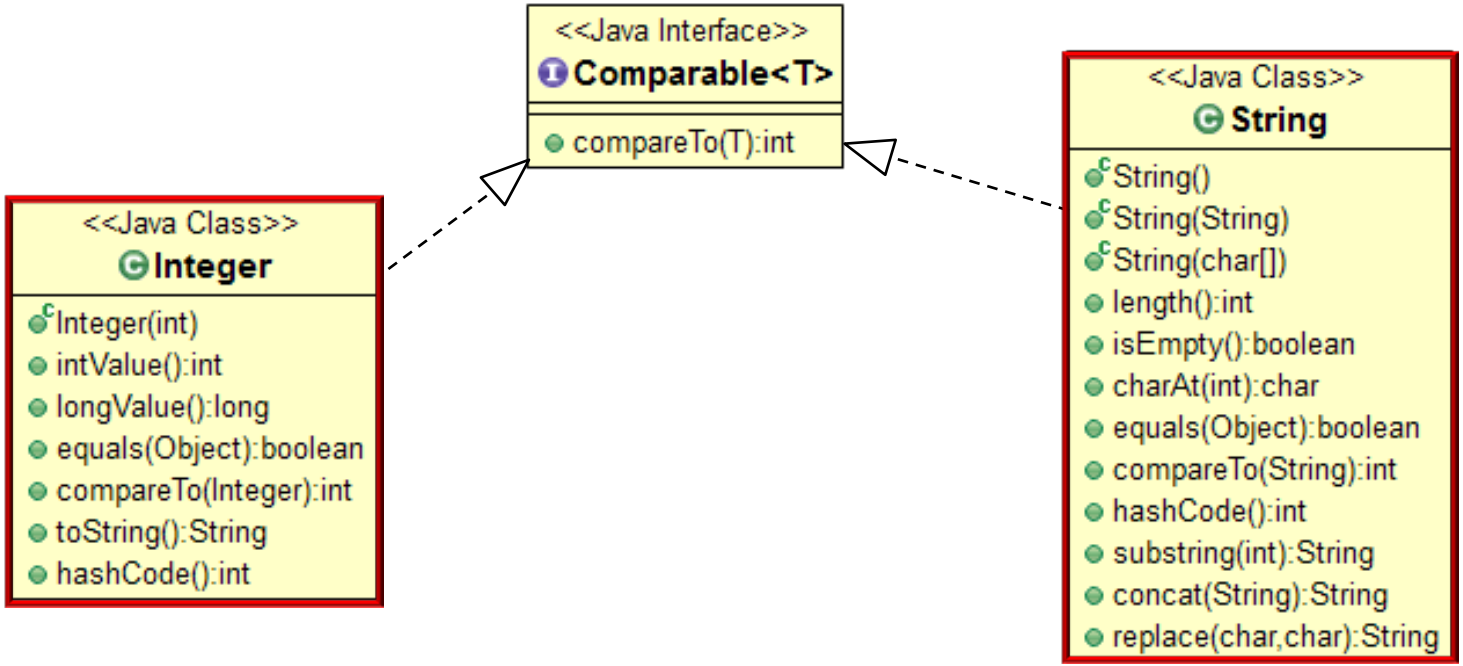
- A Java interface cannot be instantiated, but must be implemented by a class
  - The class defines the interfaces methods & any necessary fields



See developer.android.com/reference/java/lang/Comparable.html

Overview of Java's Support for Data Abstractions

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



See developer.android.com/reference/java/lang/Comparable.html

Overview of Java's Support for Data Abstractions (Part 2)

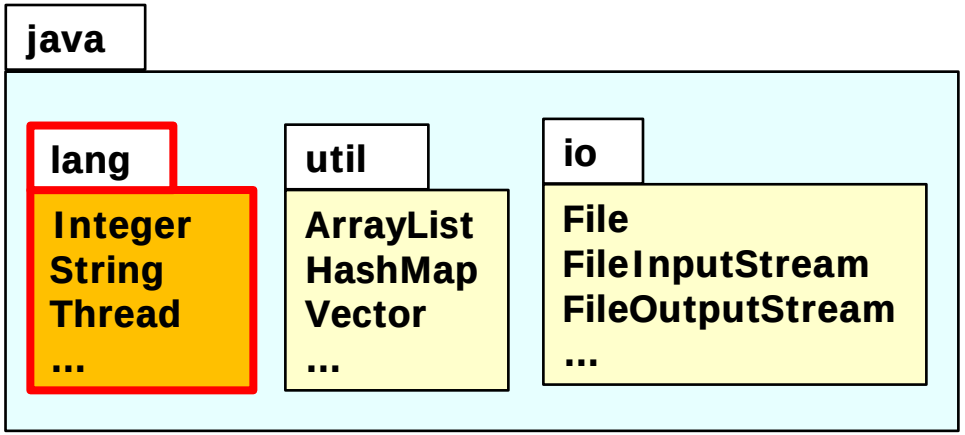
Overview of Java's Support for Data Abstractions

- Classes & interfaces can be grouped into packages

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

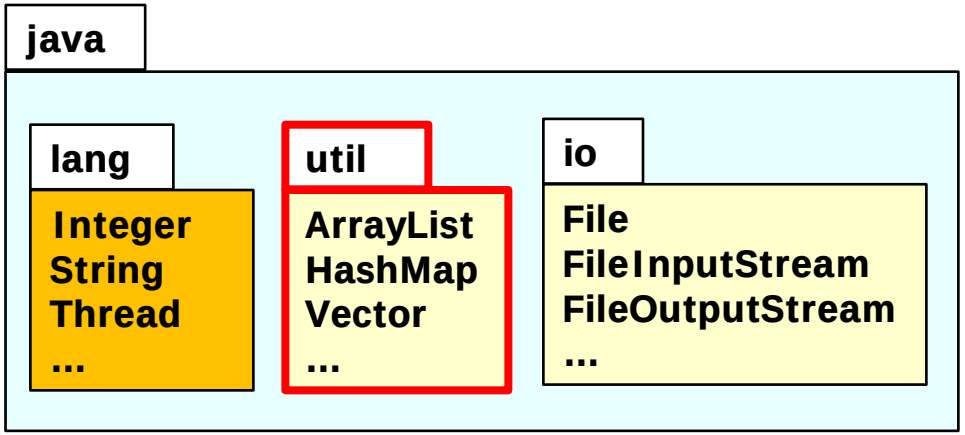
Overview of Java's Support for Data Abstractions

- Classes & interfaces can be grouped into packages, e.g.
  - java.lang contains classes fundamental to design of Java



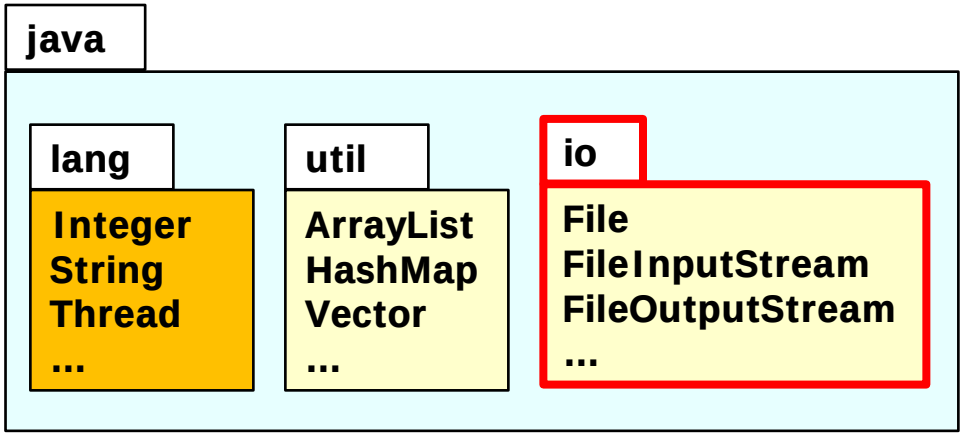
Overview of Java's Support for Data Abstractions

- Classes & interfaces can be grouped into packages, e.g.
  - java.lang contains classes fundamental to design of Java
  - java.util contains a collection of common reusable ADTs



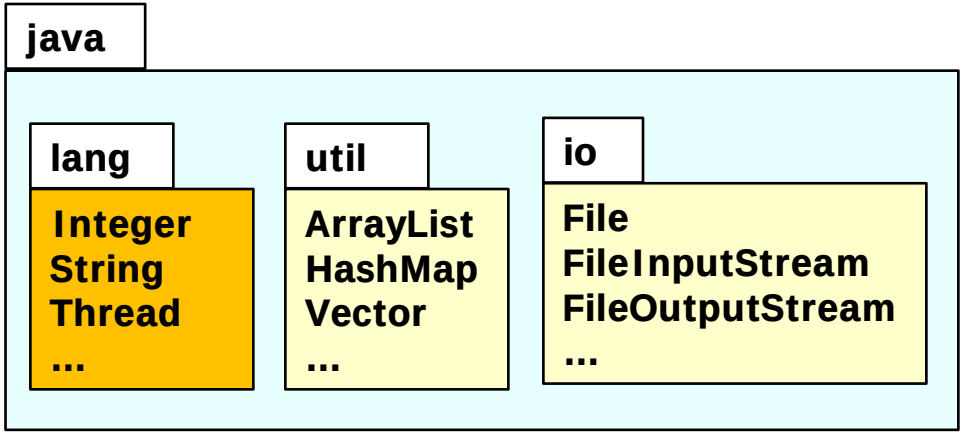
Overview of Java's Support for Data Abstractions

- Classes & interfaces can be grouped into packages, e.g.
  - java.lang contains classes fundamental to design of Java
  - java.util contains a collection of common reusable ADTs
  - java.io contains classes that provide operations on files



Overview of Java's Support for Data Abstractions

- Classes & interfaces can be grouped into packages



Packages help manage large projects by avoiding collisions for common names

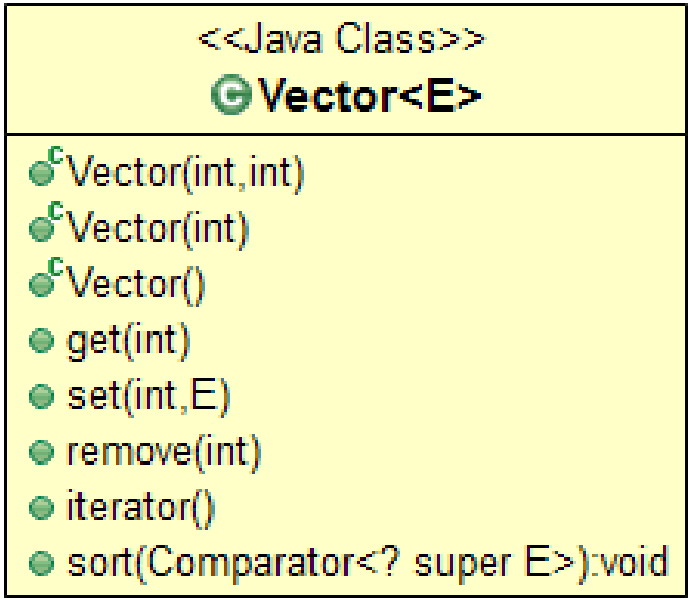
Overview of Java's Support for Data Abstractions

- Java generics enable ADTs to be parameters when defining classes, interfaces, & methods

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

Overview of Java's Support for Data Abstractions

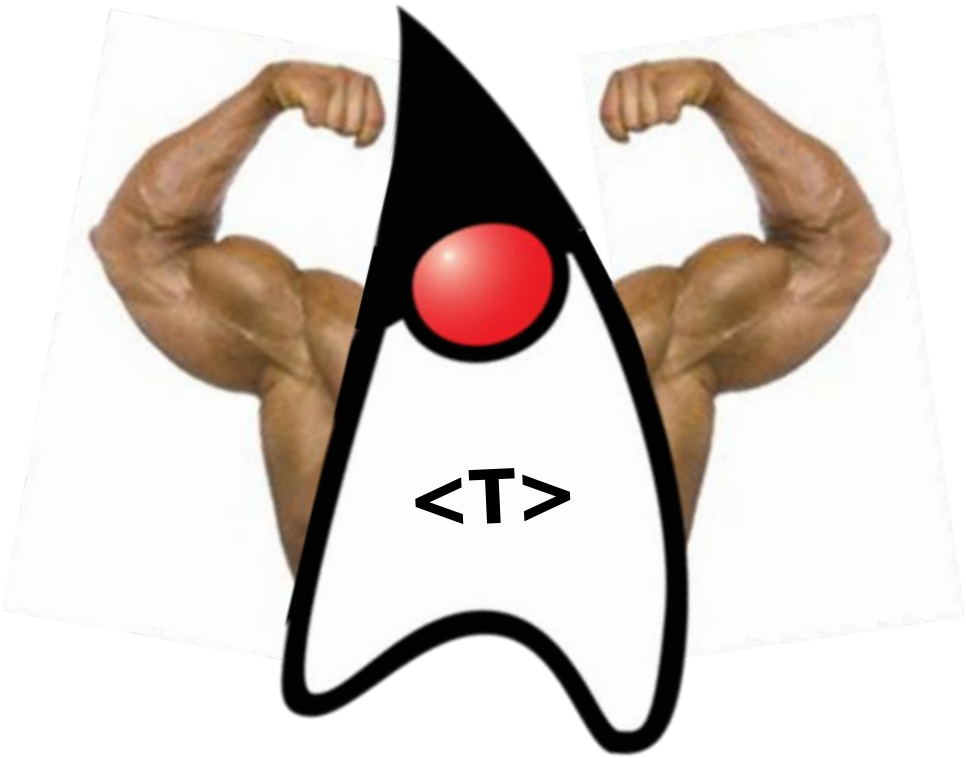
- Java generics enable ADTs to be parameters when defining classes, interfaces, & methods



See developer.android.com/reference/java/util/Vector.html

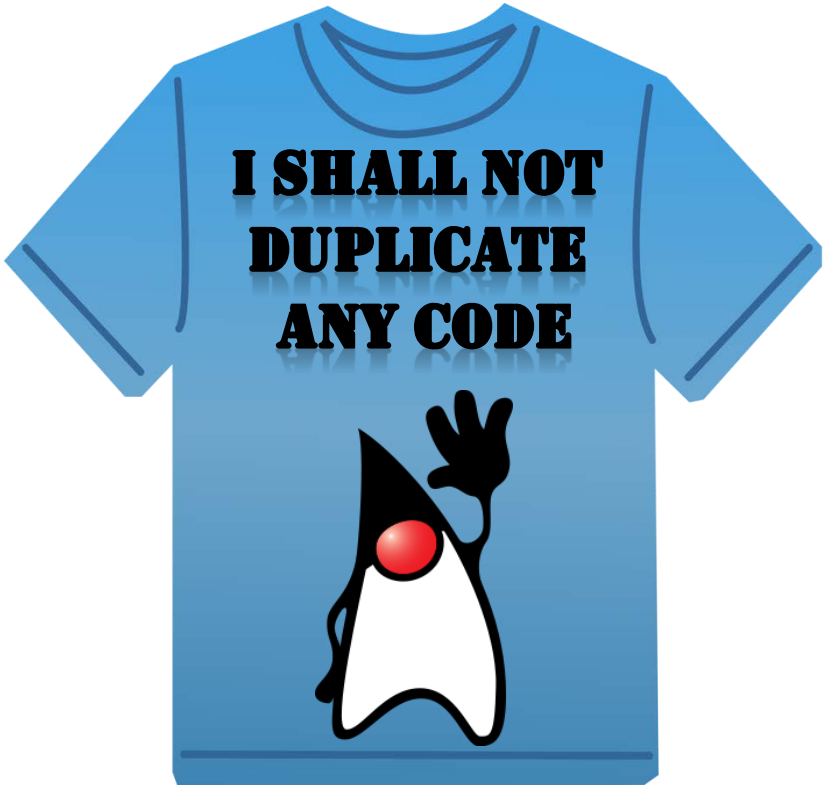
Overview of Java's Support for Data Abstractions

- Generics offer several benefits to Java programmers



Overview of Java's Support for Data Abstractions

- 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 Data Abstractions

- 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 Data Abstractions

- 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 Data Abstractions

- 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 Data Abstractions

- 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 Data Abstractions

- 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> ... {
  ...
```

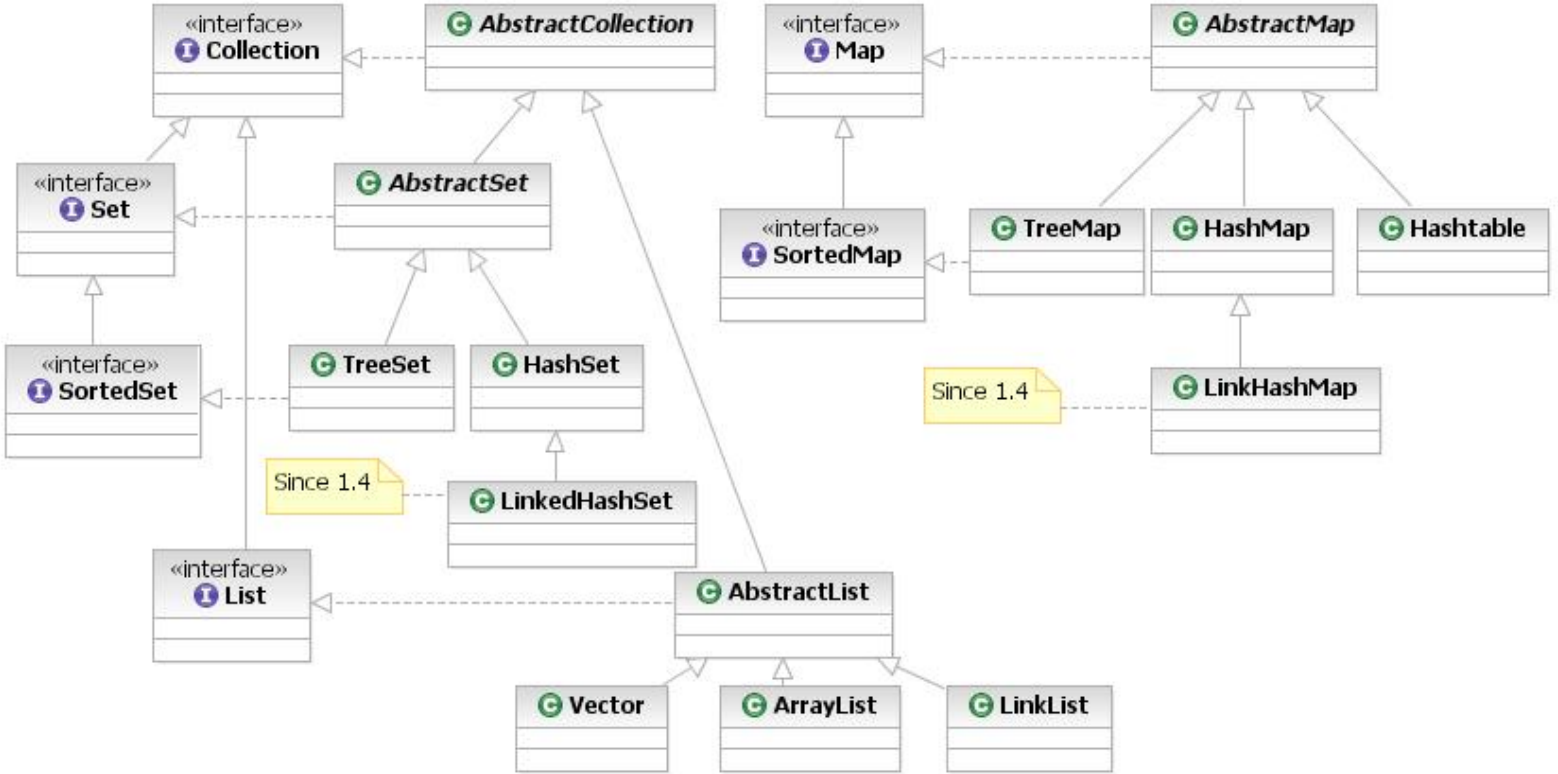
e.g.,

```
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 Data Abstractions

- The Java Collections Framework uses generic classes & interfaces extensively



See en.wikipedia.org/wiki/Java_collections_framework

Overview of Java's Support for Data Abstractions

- Java's garbage collector automatically reclaims & recycles memory that is not in use by a program

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

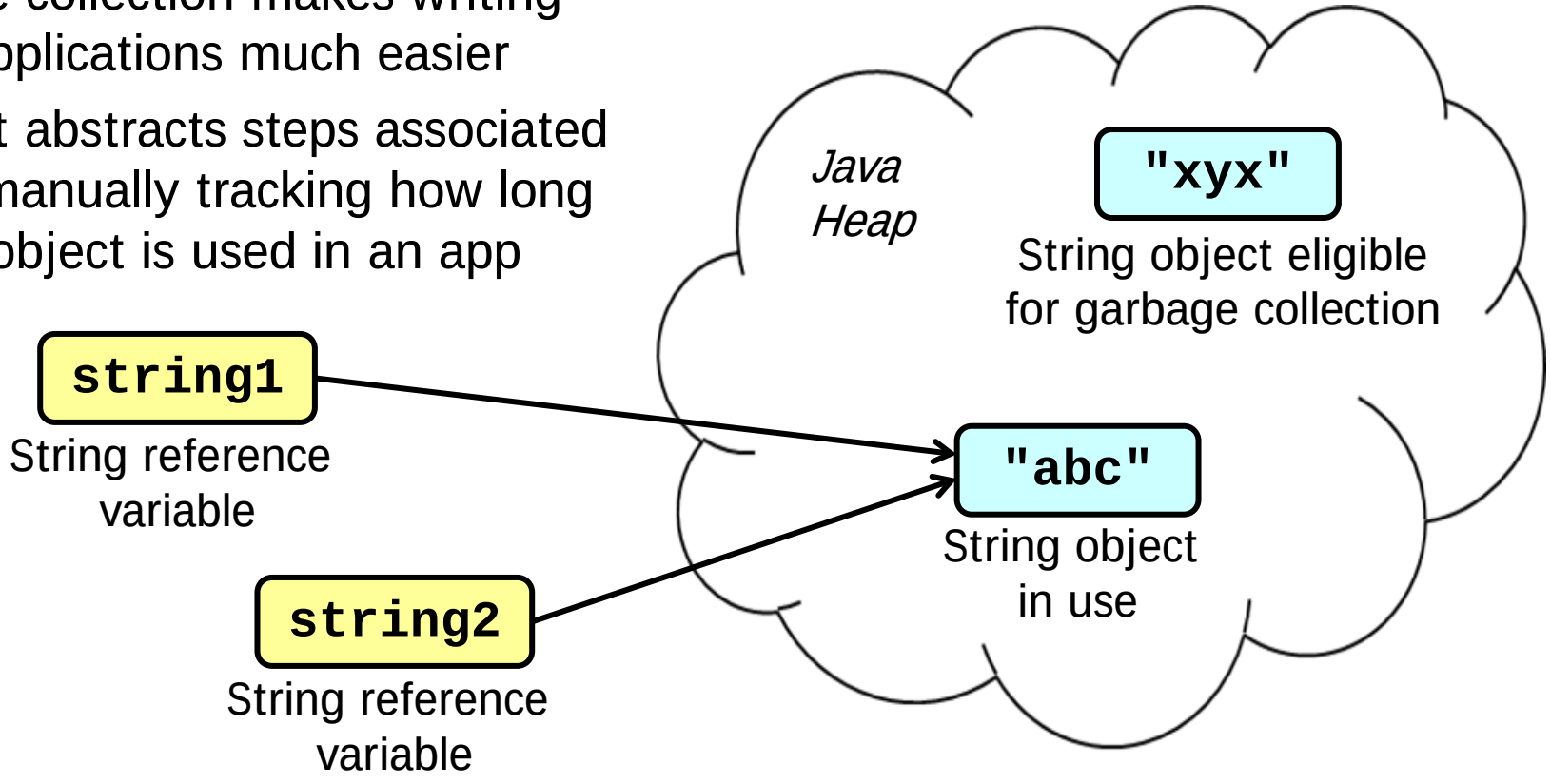
Overview of Java's Support for Data Abstractions

- Garbage collection makes writing many applications much easier

See en.wikipedia.org/wiki/Garbage_collection_(computer_science)#Advantages

Overview of Java's Support for Data Abstractions

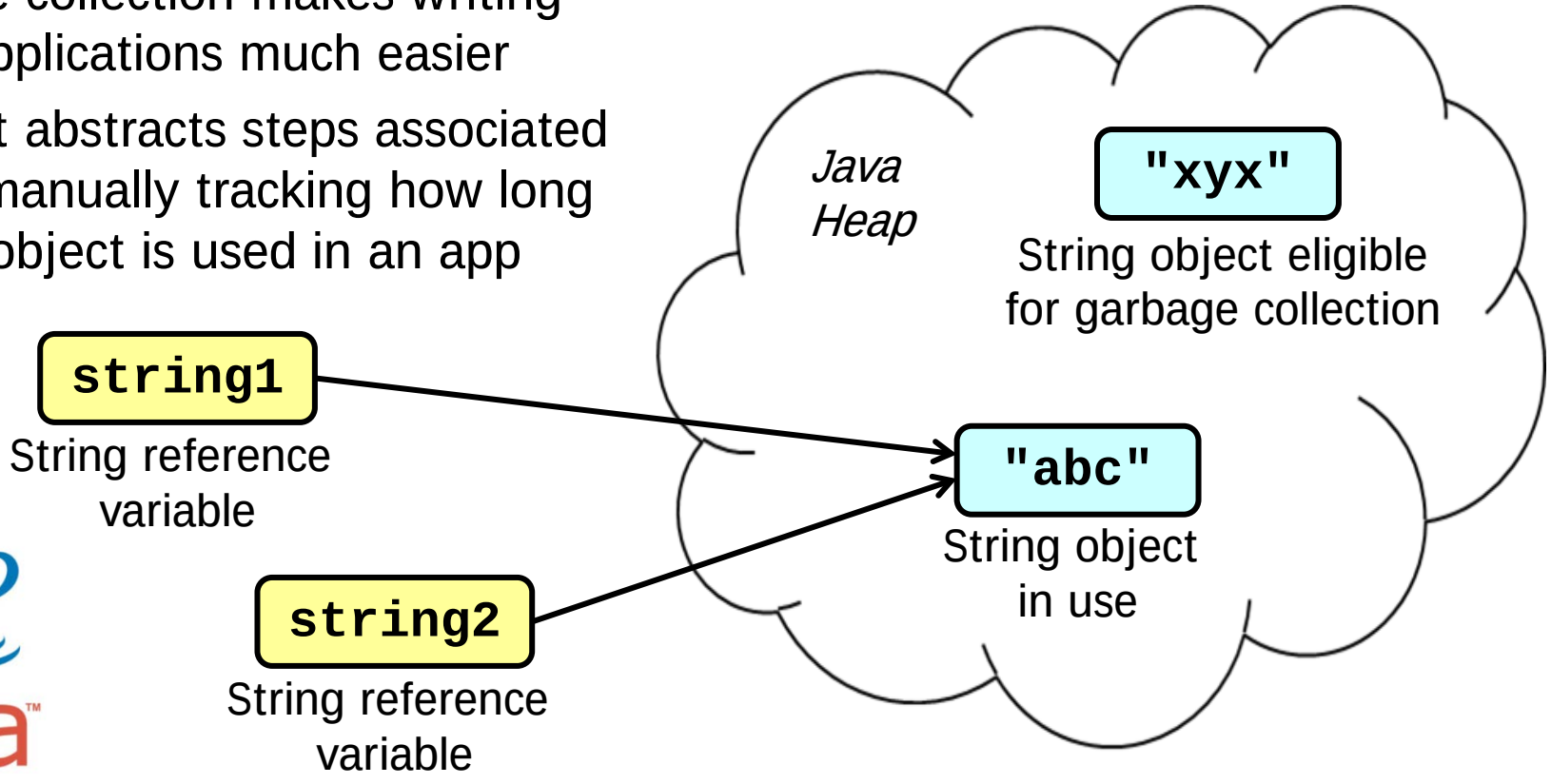
- Garbage collection makes writing many applications much easier
  - e.g., it abstracts steps associated with manually tracking how long each object is used in an app



Not all object-oriented languages support garbage collection...

Overview of Java's Support for Data Abstractions

- Garbage collection makes writing many applications much easier
  - e.g., it abstracts steps associated with manually tracking how long each object is used in an app

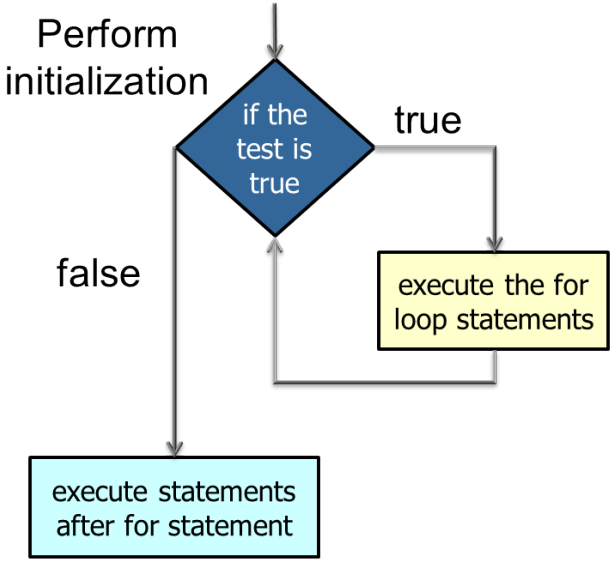


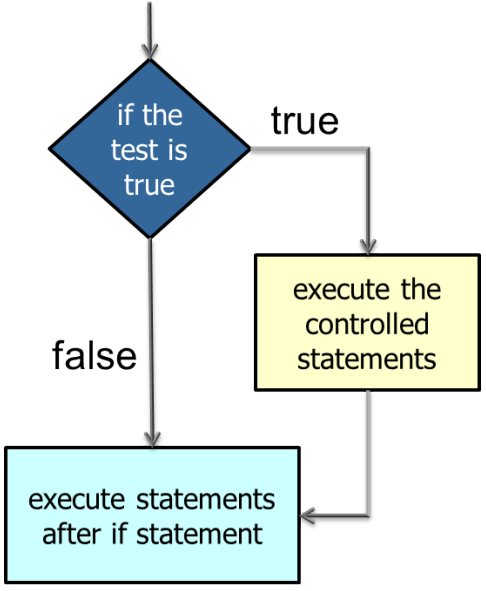
... but garbage collection is an essential feature in Java

Overview of Java's Support for Control Abstractions

Overview of Java's Support for Control Abstractions

- Java supports several control abstractions

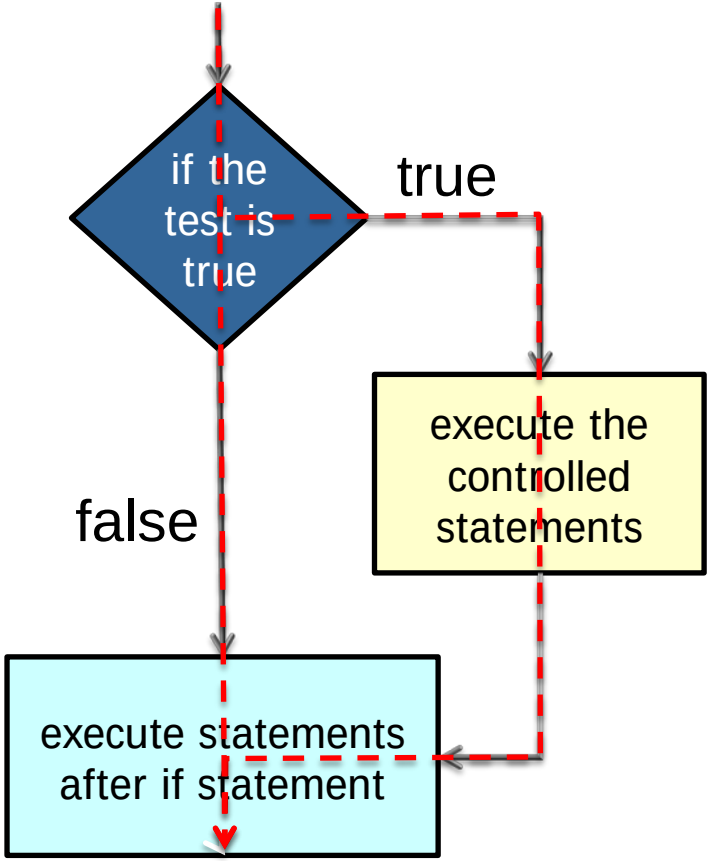




See en.wikipedia.org/wiki/Abstraction_(computer_science)#Control_abstraction

Overview of Java's Support for Control Abstractions

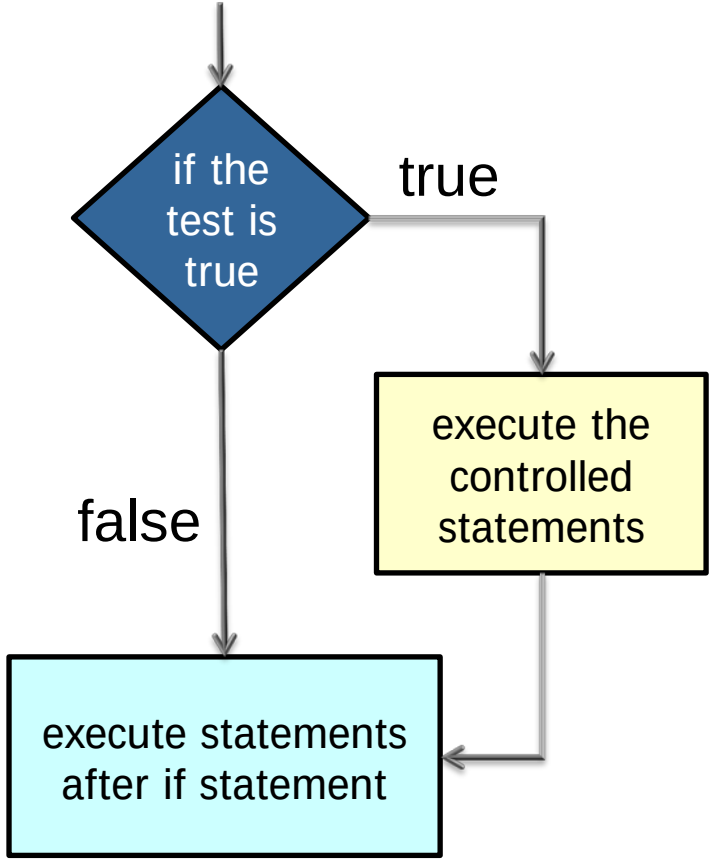
- Java conditional statements can be used to selectively alter program control flow



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

Overview of Java's Support for Control Abstractions

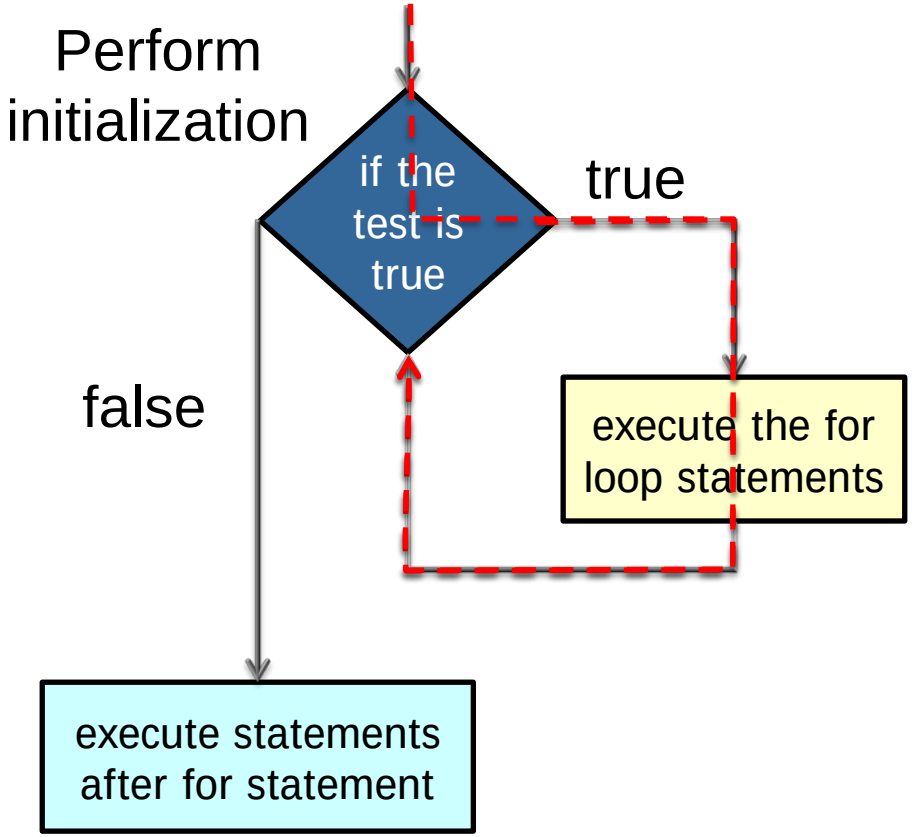
- Java conditional statements can be used to selectively alter program control flow
  - e.g., if/else statement



See docs.oracle.com/javase/tutorial/java/nutsandbolts/if.html

Overview of Java's Support for Control Abstractions

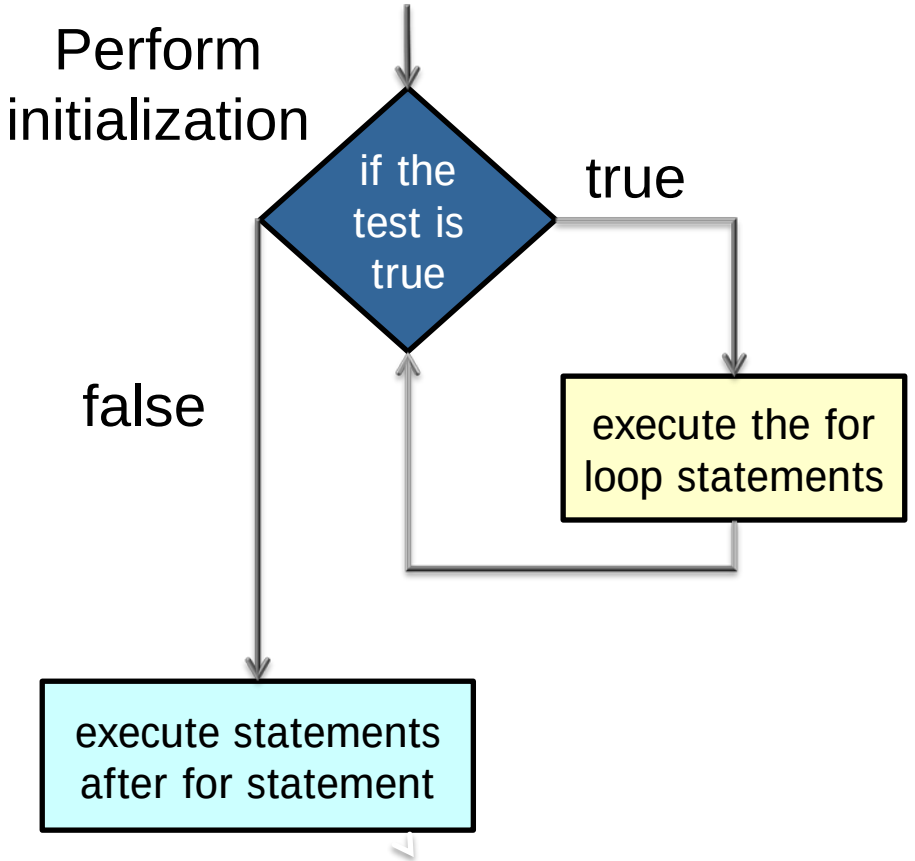
- Java iteration features enable repetition of a block of one or more statements



See en.wikipedia.org/wiki/Iteration#Computing

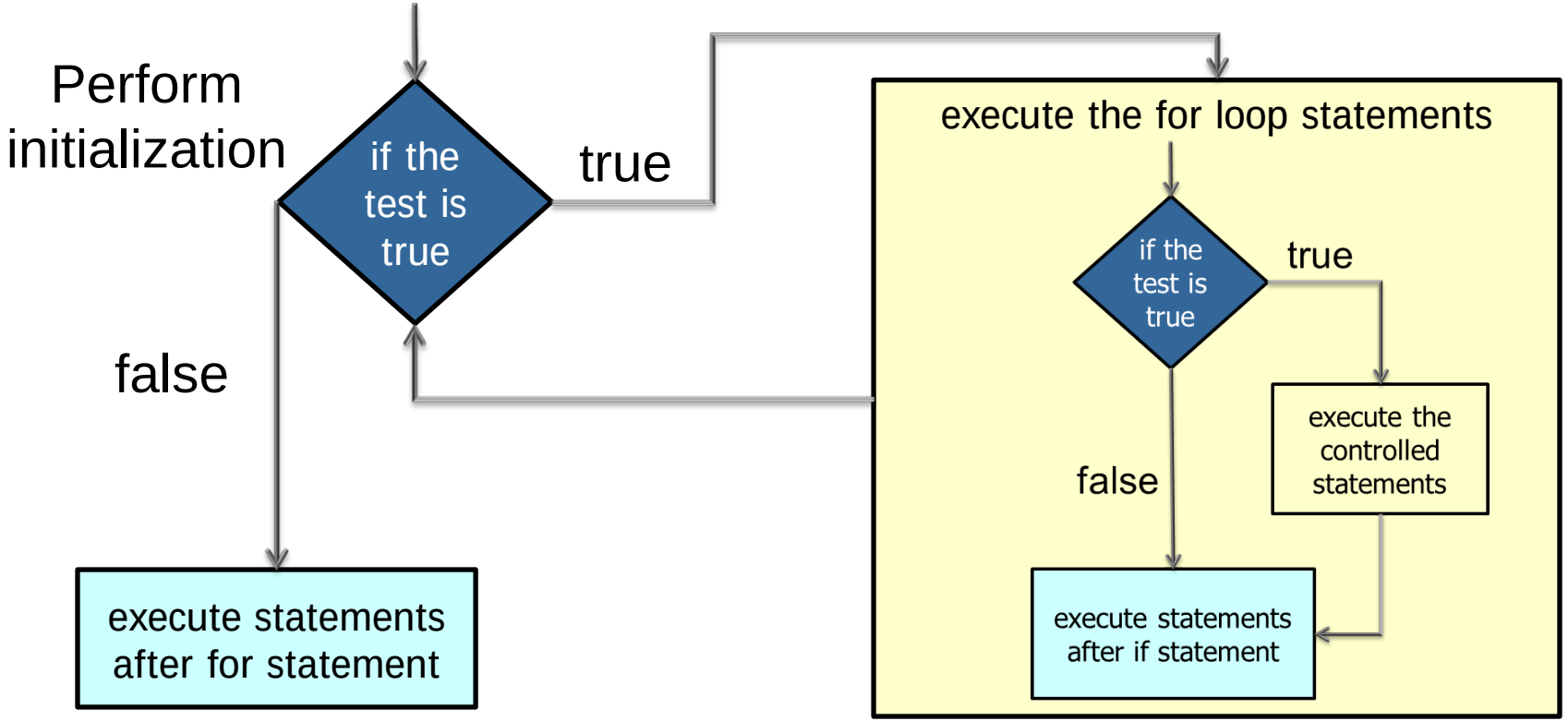
Overview of Java's Support for Control Abstractions

- Java iteration features enable repetition of a block of one or more statements
  - e.g., for loop, while loop, & do/while loop



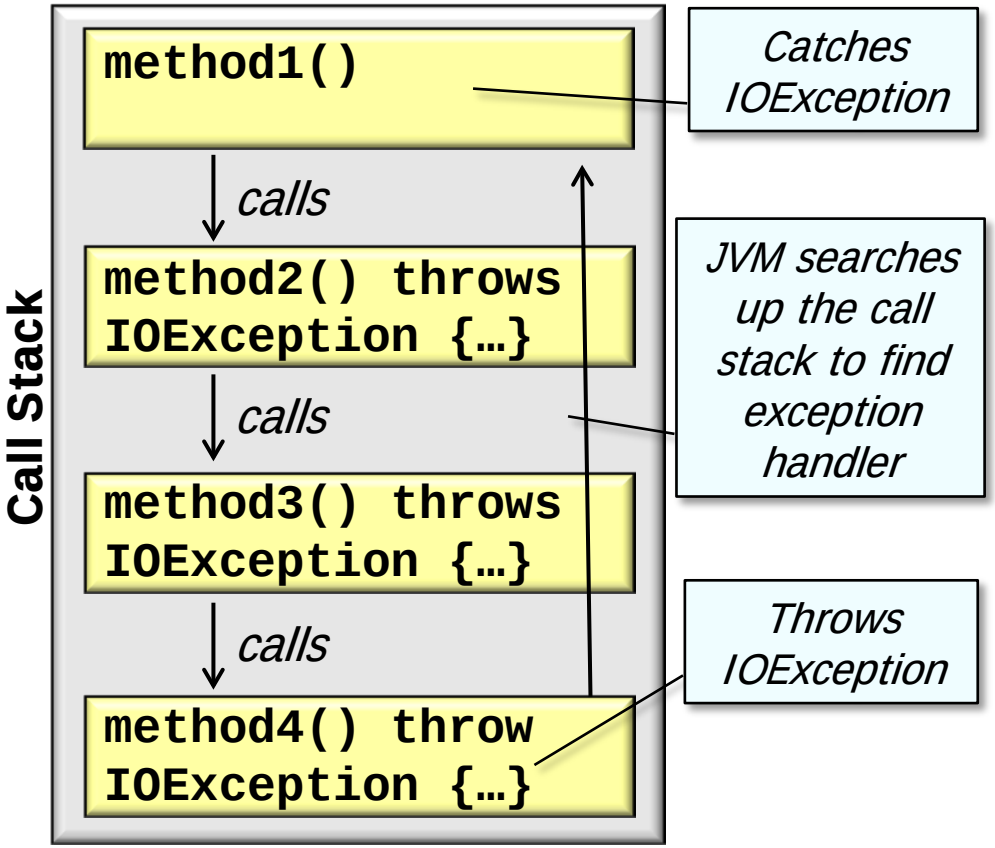
Overview of Java's Support for Control Abstractions

- Iteration features often combine with conditional statements in Java apps



Overview of Java's Support for Control Abstractions

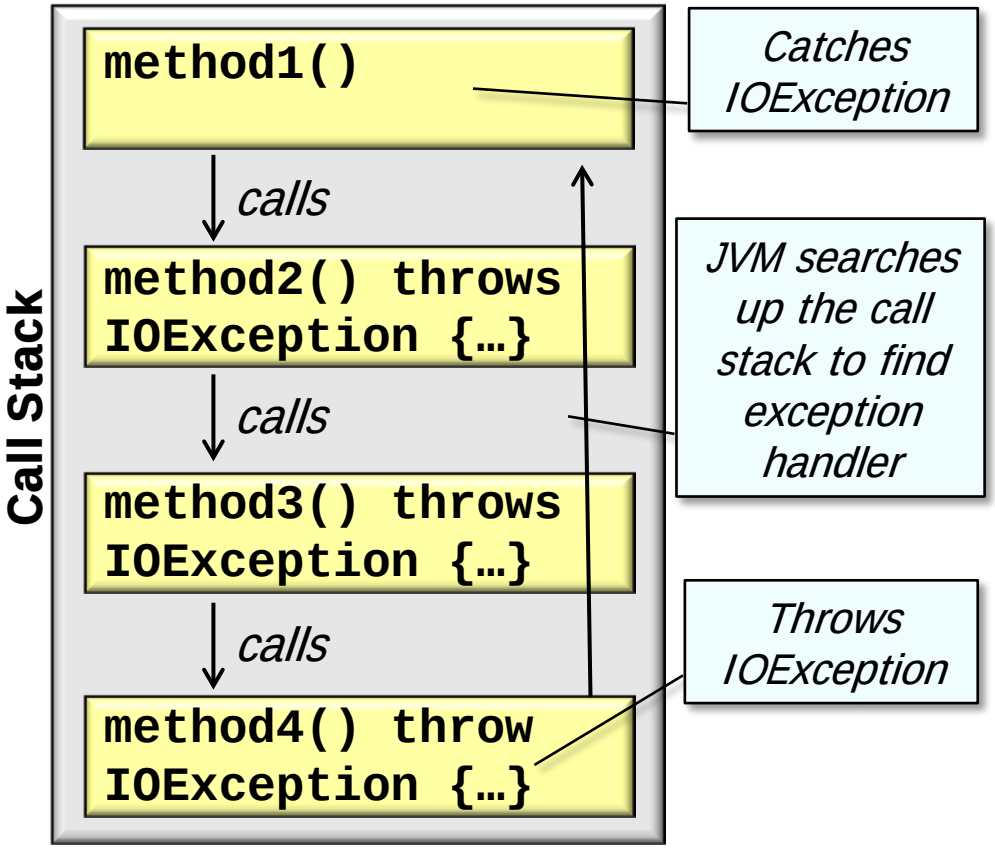
- Java exception handling separates "normal" app execution from "anomalous" app execution



See en.wikipedia.org/wiki/Exception_handling

Overview of Java's Support for Control Abstractions

- Java exception handling separates "normal" app execution from "anomalous" app execution



Exception handling makes Java apps more robust, understandable, & evolvable