OBJECTS AND CLASSES

ASHESI

Recall: Object-Oriented Programming

- Basic idea:
  - The real world is made up of objects
    - E.g. People, cars, buildings, politicians, countries, tomatoes
  - These objects have attributes
    - E.g. a person has a name, an age, a temperature, a heart rate, etc…
    - E.g. A car has a make, a model, a color, some fuel, etc…
  - These objects each have the ability to perform certain actions/behaviors, which may affect themselves and/or other objects
    - E.g. A person can grow, a person can earn money, a person can marry another person, a person can drive a car, a person can eat a tomato etc…
    - E.g. A politician can give a speech, a politician may become the president of a country, a politician may be indicted for corruption etc…
  - In object-oriented programming, we consider a program to similarly consist of objects that can act alone or interact with each other

ASHESI

Recall: Objects and Classes

- In object-oriented programming, an object represents a real-world object or abstraction
  - E.g. a vehicle, a student, a bank account
- Objects of the same kind are said to belong to the same class – they have the same data type
- A class is a plan or blueprint for creating objects of a particular type. It specifies:
  - *Data* or *attributes* that objects of the class have
  - Also *methods* or *actions* that these objects can take

ASHESI

Recall: Objects and Classes

- An object is an instance of a class
- Objects of the same class have the same attributes and behaviors/actions but are not necessarily identical
  - E.g. All people have a name and an age, but not the **same** name and age. All people can grow and eat
  - We can define a Person class
    - Attributes (data): name, age
    - Behavior/actions (methods): grow, eat
  - We can create some objects or instances of the Person class
    - Person 1: name: Ama, age: 5
    - Person 2: name: Kofi, age: 33
    - Person 3: name: Ebo, age: 12

ASHESI

Java Built-In Classes

- We have already used some pre-defined or built-in classes:
  - String class
    - Data: A sequence of characters
    - Methods: length(), charAt(), substring(), equals(), etc…
    - Example: creating String objects:

```
String name1 = "Ayorkor";
String name2 = new String("Joe");
```

    - Example: invoking the behaviors/actions (methods) of String objects:

```
int len = name1.length();
char firstLetter = name1.charAt(0);
```

  - Scanner class
    - Data:
    - Methods: next(), nextInt(), nextDouble(), etc…
    - Example: creating a Scanner object:

```
Scanner input = new Scanner(System.in);
```

    - Example: invoking the behaviors/actions (methods) of a Scanner object:

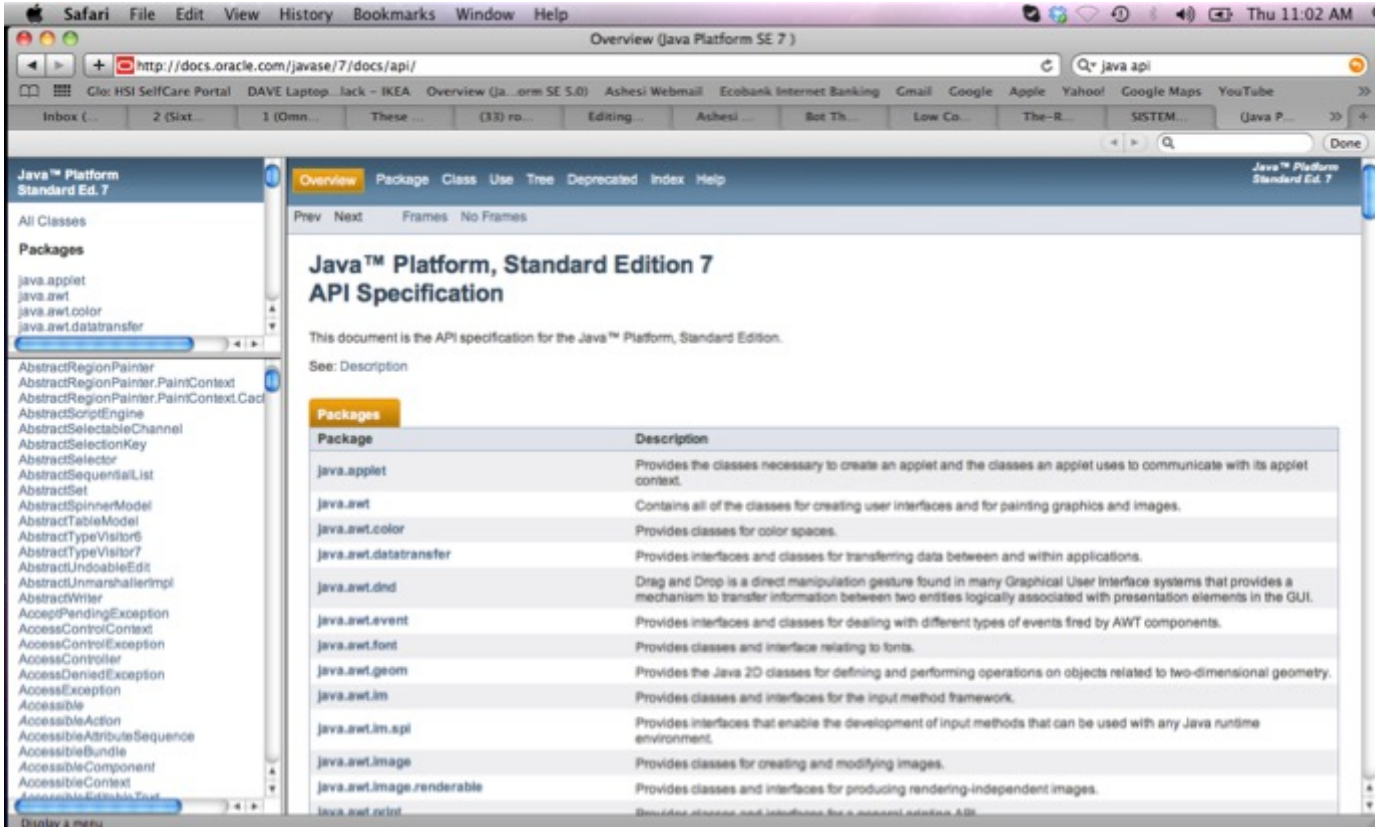
```
double number = input.nextDouble();
```

The keyword `new` is used to create / instantiate objects of a class

ASHESI

Java Built-In Classes

- Documentation of all Java built-in classes: Java API Specification: http://docs.oracle.com/javase/7/docs/api/
  - Note: API stands for “Application Programming Interface”



ASHESI

Java API: String class

Java™ Platform
Standard Ed. 7

All Classes

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer

StAXResult
StAXSource
Streamable
StreamableValue
StreamCorruptedException
StreamFilter
StreamHandler
StreamPrintService
StreamPrintServiceFactory
StreamReaderDelegate
StreamResult
StreamSource
StreamTokenizer
StrictMath
String
StringBuffer
StringBufferInputStream
StringBuilder
StringCharacterIterator
StringContent
StringHolder
StringIndexOutOfBoundsException
StringMonitor
StringMonitorMBean
StringNameHelper
StringReader
StringRefAddr
StringSelection
StringSeqHelper

Overview Package Class Use Tree Deprecated Index Help

Java™ Platform Standard Ed. 7

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.lang

Class String

java.lang.Object
  java.lang.String

All Implemented Interfaces:

Serializable, CharSequence, Comparable<String>

```
public final class String
extends Object
implements Serializable, Comparable<String>, CharSequence
```

The `String` class represents character strings. All string literals in Java programs, such as `"abc"`, are implemented as instances of this class.

Strings are constant; their values cannot be changed after they are created. String buffers support mutable strings. Because String objects are immutable they can be shared. For example:

```
String str = "abc";
```

is equivalent to:

```
char data[] = {'a', 'b', 'c'};
String str = new String(data);
```

Here are some more examples of how strings can be used:

```
System.out.println("abc");
String cde = "cde";
System.out.println("abc" + cde);
```

ASHESI

Java API: String class

Java™ Platform
Standard Ed. 7

All Classes

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer

StAXResult
StAXSource
Streamable
StreamableValue
StreamCorruptedException
StreamFilter
StreamHandler
StreamPrintService
StreamPrintServiceFactory
StreamReaderDelegate
StreamResult
StreamSource
StreamTokenizer
StrictMath
String
StringBuffer
StringBufferInputStream
StringBuilder
StringCharacterIterator
StringContent
StringHolder
StringIndexOutOfBoundsException
StringMonitor
StringMonitorMBean
StringNameHelper
StringReader
StringRefAddr
StringSelection
StringSegHelper

Method Summary

Methods

Modifier and Type	Method and Description
`char`	`charAt(int index)` Returns the `char` value at the specified index.
`int`	`codePointAt(int index)` Returns the character (Unicode code point) at the specified index.
`int`	`codePointBefore(int index)` Returns the character (Unicode code point) before the specified index.
`int`	`codePointCount(int beginIndex, int endIndex)` Returns the number of Unicode code points in the specified text range of this `String`.
`int`	`compareTo(String anotherString)` Compares two strings lexicographically.
`int`	`compareToIgnoreCase(String str)` Compares two strings lexicographically, ignoring case differences.
`String`	`concat(String str)` Concatenates the specified string to the end of this string.
`boolean`	`contains(CharSequence s)` Returns true if and only if this string contains the specified sequence of char values.
`boolean`	`contentEquals(CharSequence cs)` Compares this string to the specified `CharSequence`.
`boolean`	`contentEquals(StringBuffer sb)` Compares this string to the specified `StringBuffer`.
`static String`	`copyValueOf(char[] data)` Returns a String that represents the character sequence in the array specified.
`static String`	`copyValueOf(char[] data, int offset, int count)` Returns a String that represents the character sequence in the array specified.
`boolean`	`endsWith(String suffix)` Tests if this string ends with the specified suffix.
`boolean`	`equals(Object anObject)`

ASHESI

Java API: Scanner class

Java™ Platform
Standard Ed. 7

All Classes

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer

SaslClientFactory
SaslException
SaslServer
SaslServerFactory
Savepoint
SAXException
SAXNotRecognizedException
SAXNotSupportedException
SAXParseException
SAXParser
SAXParserFactory
SAXResult
SAXSource
SAXTransformerFactory
Scanner
ScatteringByteChannel
ScheduledExecutorService
ScheduledFuture
ScheduledThreadPoolExecutor
Schema
SchemaFactory
SchemaFactoryLoader
SchemaOutputResolver
SchemaViolationException
ScriptContext
ScriptEngine
ScriptEngineFactory
ScriptEngineManager
ScriptException

Overview Package Class Use Tree Deprecated Index Help

Java™ Platform
Standard Ed. 7

Prev Class Next Class Frames No Frames

Summary: Nested | Field | Constr | Method Detail: Field | Constr | Method

java.util

Class Scanner

java.lang.Object
  java.util.Scanner

All Implemented Interfaces:

Closeable, AutoCloseable, Iterator<String>

```
public final class Scanner
extends Object
implements Iterator<String>, Closeable
```

A simple text scanner which can parse primitive types and strings using regular expressions.

A `Scanner` breaks its input into tokens using a delimiter pattern, which by default matches whitespace. The resulting tokens may then be converted into values of different types using the various `next` methods.

For example, this code allows a user to read a number from `System.in`:

```
Scanner sc = new Scanner(System.in);
int i = sc.nextInt();
```

As another example, this code allows `long` types to be assigned from entries in a file `myNumbers`:

```
Scanner sc = new Scanner(new File("myNumbers"));
while (sc.hasNextLong()) {
    long aLong = sc.nextLong();
}
```

The scanner can also use delimiters other than whitespace. This example reads several items in from a string:

ASHESI

Java API: Scanner Class

Java™ Platform
Standard Ed. 7

All Classes

Packages

java.applet
java.awt
java.awt.color
java.awt.datatransfer

SaslClientFactory
SaslException
SaslServer
SaslServerFactory
Savepoint
SAXException
SAXNotRecognizedException
SAXNotSupportedException
SAXParseException
SAXParser
SAXParserFactory
SAXResult
SAXSource
SAXTransformerFactory
Scanner
ScatteringByteChannel
ScheduledExecutorService
ScheduledFuture
ScheduledThreadPoolExecutor
Schema
SchemaFactory
SchemaFactoryLoader
SchemaOutputResolver
SchemaViolationException
ScriptContext
ScriptEngine
ScriptEngineFactory
ScriptEngineManager
ScriptException

Method Summary

Methods

Modifier and Type	Method and Description
void	`close()` Closes this scanner.
`Pattern`	`delimiter()` Returns the `Pattern` this `Scanner` is currently using to match delimiters.
`String`	`findInLine(Pattern pattern)` Attempts to find the next occurrence of the specified pattern ignoring delimiters.
`String`	`findInLine(String pattern)` Attempts to find the next occurrence of a pattern constructed from the specified string, ignoring delimiters.
`String`	`findWithinHorizon(Pattern pattern, int horizon)` Attempts to find the next occurrence of the specified pattern.
`String`	`findWithinHorizon(String pattern, int horizon)` Attempts to find the next occurrence of a pattern constructed from the specified string, ignoring delimiters.
boolean	`hasNext()` Returns true if this scanner has another token in its input.
boolean	`hasNext(Pattern pattern)` Returns true if the next complete token matches the specified pattern.
boolean	`hasNext(String pattern)` Returns true if the next token matches the pattern constructed from the specified string.
boolean	`hasNextBigDecimal()` Returns true if the next token in this scanner's input can be interpreted as a `BigDecimal` using the `nextBigDecimal()` method.
boolean	`hasNextBigInteger()` Returns true if the next token in this scanner's input can be interpreted as a `BigInteger` in the default radix using the `nextBigInteger()` method.
boolean	`hasNextBigInteger(int radix)` Returns true if the next token in this scanner's input can be interpreted as a `BigInteger` in the specified radix using the `nextBigInteger()` method.

ASHESI

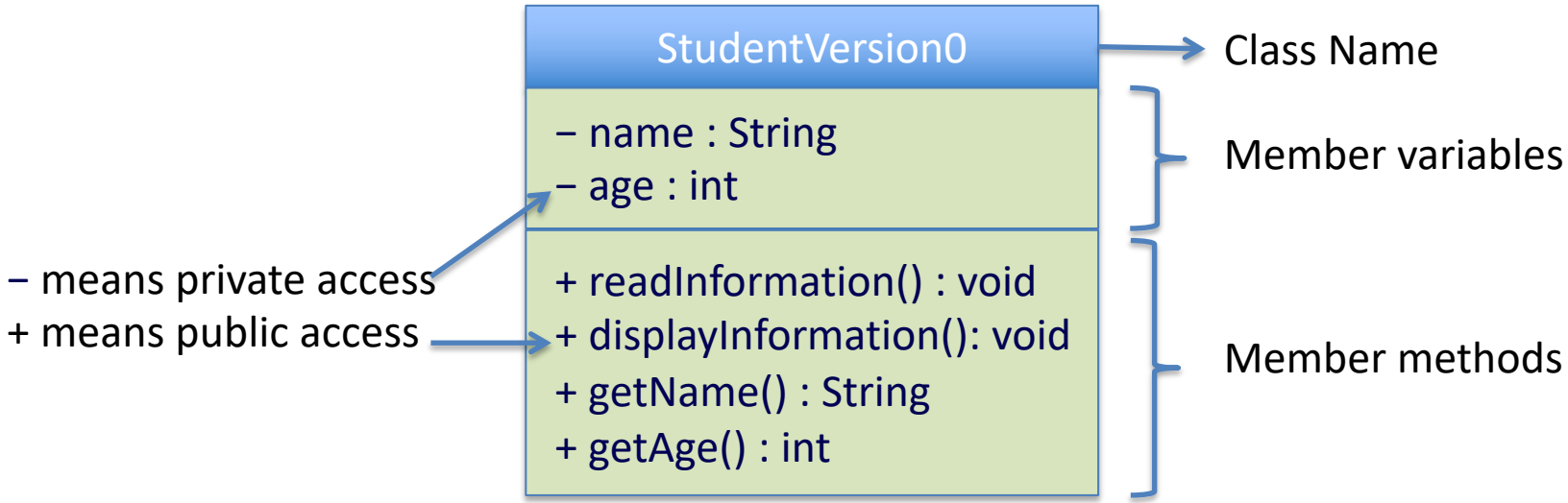
Designing and Implementing Classes

- We can create our own classes!
- We can create a simple class to represent a student. The code is in StudentVersion0.java
  - Member variables:
    - String name, int age
  - Methods:
    - readInformation(), displayInformation(), getName(), getAge()
- A test program (a main method) is in TestStudentVersion0.java
- We could also have put the main method in the same class. See StudentVersion0b.java

ASHESI

UML Diagrams

- UML Diagrams represent the design of classes
- A UML diagram for our StudentVersion0 class:



- See StudentVersion0.java for the implementation of the class, and TestStudentVersion0.java for a test program

ASHESI