CS 113 – Computer Science I

Lecture 11 – Objects

Adam Poliak
10/20/2022
Announcements

• Assignment 05
  • Due Thursday 10/20 - tonight

• Sharing code
Data types revisited

What are some examples of built-in types in Java?

What is a data type?
Examples

<table>
<thead>
<tr>
<th>Type</th>
<th>Valid values</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Examples

<table>
<thead>
<tr>
<th>Type</th>
<th>Valid values</th>
<th>Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>1, 10, 999</td>
<td>%, +, -, / ...</td>
</tr>
<tr>
<td>boolean</td>
<td>true, false</td>
<td>==, &amp;&amp;,</td>
</tr>
<tr>
<td>String</td>
<td>Anything between &quot;&quot;</td>
<td>.compareTo(), .charAt(), concatenation, ...</td>
</tr>
</tbody>
</table>
Classes and objects

An **object** is to a **class** as a

- **cat** is to an **animal**
- **tulip** is to an **flower**
- **cookie** is to a **snack**
- Socrates is to a **human**
Classes and objects

A **class** defines the characteristics of a type (data and methods)

An **object** is a particular example of a class

Java is a strict object-oriented programming language, meaning all code must be inside a class!
Creating objects

Declare variables in the same way!

Create using `new`
Using objects

The methods you are allowed to call on an object is called an **API**

Recall: API = Application Programming Interface

Example: The *String API* has over 60 methods!

Objects can have either *static* or *instance* methods

- static methods use syntax `<ClassName>..<methodName>`
- instance methods use syntax `<object>..<methodName>`
Example: String API

<table>
<thead>
<tr>
<th>boolean</th>
<th>endsWith(String suffix)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Tests if this string ends with the specified suffix.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>equals(Object anObject)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compares this string to the specified object.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>boolean</th>
<th>equalsIgnoreCase(String anotherString)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Compares this String to another String, ignoring case considerations.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static String</th>
<th>format(Locale l, String format, Object... args)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns a formatted string using the specified locale, format string, and arguments.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>static String</th>
<th>format(String format, Object... args)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Returns a formatted string using the specified format string and arguments.</td>
</tr>
</tbody>
</table>
Example: String API
Using objects: some special methods

The **constructor method** is called when you do a `new`.

**accessors (aka getters)**
return the values of instance variables

**mutators (aka setters)**
set the values of instance variables

**toString()**
returns a string representation of an object
Defining classes

By defining our own classes, we can create our own data types

A class definition contains

- the data contained by the new type (instance variables)

- the operations supported by the new type (instance methods)
Example: Defining a class `Point`

What data should it have?

What operations should it support?
Object-oriented programming (OOP)

Method for designing programs in terms of objects

Recall: Top-down design

- the “nouns” in your feature list correspond to classes/data
- the “verbs” correspond to methods
OOP Example & Design: Vending machine
OOP Design: Vending machine
public class Snack {
    private int mQuantity;
    private double mCost;
    private String mName;

    public Snack(String name, int quantity, double cost) {
        mQuantity = quantity;
        mCost = cost;
        mName = name;
    }

    public String getName() {
        return mName;
    }

    public void buy() {
        if (mQuantity > 0) {
            mQuantity--;
        }
    }
}
Testing the Snack class

```java
public static void main(String args[])
{
    Snack snack = new Snack("Slurm", 10, 1.5);
    System.out.println("Snack: " + snack.getName());
}
```
Objects: Stack diagrams revisited

```java
public static void main(String[] args) {
    double userCash = 8.0;
    Snack soda = new Snack("Tang", 10, 1.5); // call constructor
    soda.buy();
}
```
Exercise: draw a stack diagram for this program
Exercise: Define a class BankAccount

BankAccount should have the following data:

- Name
- Amount

BankAccount should have the following operations:

- `currentBalance()` // returns current amount in the bank account
- `withdraw(float amt)` // withdraw the given amount from the account
- `deposit(float amt)` // deposit the given amount to the account