**Good to know**

* In case of ambiguities you can always check out the Udacity web course. It explains details of android development really good.
* Open GL ES is a native C/C++ library that handles pictures and videos (not used)
* To remove the whole folder and subsequent files, use **rm –rf mydir**
* You can set the theme so that your application/service looks like a pop-up or etc.
* To add to gravity you can use the following syntax (**hahahaha**)
  + android:gravity="center|bottom"

**Java**

* In Java errors are called exceptions (**hahahaha**)
* **ForEach in Java**

package com.example.java;

public class Main{

String [] months = { “January”, “February”, “March”, …,”December”};

for( String month : months){

System.out.println(month);

}

**Lynda**

**4 components of an Android program**

1. **Activities**
   1. A single screen with a UI (List of emails is an activity)
2. **Services** 
   1. Components that run for a long time
   2. It can run in the background.
   3. It can bind into other applications
3. **Content Provider**
   1. The component responsible for sharing the data between applications
4. **Broadcast Receivers** 
   1. A component that responds to system-wide broadcast announcements
   2. Ex. Screen turns off, battery is low
   3. No user interface but you can have a notification

**Android Manifest File**

1. The root directory of your applications that contains all the permissions, and the minimum API required for the user to see the application on the play store or to be able to run it

**Checking for the optimization**

1. You can search android sdk folder and open up the hierarchy viewer and run the processes on the phone/emulator and see how optimized each activity is (so far I have not been able to locate that folder)
2. You can also use traceview from the google developer website to optimize the app

**The Keyword “This”**

* Refers to the current instance of sth, mainly activities. For instance you can have MainActivity.this, which refers to the current instance of the activity

**Override**

* It is an annotation mainly to show that your class declaration overrides the superclass. If you don’t include the @override, you will receive a warning. However, if you use it but don’t have a superclass you will get an error

**New**

* To declare a class in java you need two parts. First the declaration and naming the class. Second, we need a physical copy of the variable (class) to be initialized and space being allocated for it.
* In second step, we can use the new operator. It is a dynamically allocated memory for a new object and returning a reference to the main program
* The reference then is stored in a new variable
* In Java, all the classes must be dynamically allocated.
  + Example
    - public class Box {

double width;

double height;

double length;

}

.

.

.

Box mybox = new Box();

Double height = new Box().height

**Static**

* It is a way of grouping classes together. Java does not allow you to create top-level statics, only nested or inner classes.
* Example

**public class CarParts {**

**public static class Wheel {**

**public Wheel() {**

**System.out.println("Wheel created!");**

**}**

**}**

**public CarParts() {**

**System.out.println("Car Parts object created!");**

**}**

**}**

Remember that declaring the CarParts() does not create Wheel.

**Intent**

* It is a simple message object that is used to communicate between android components such as activities, content providers, broadcast receivers, and services.
* You can use intent to
  + Launch an activity
  + Start a New service
  + For Broadcasting Messages
  + To Display a list of contacts in ListView
* **2 Types**
  + **Implicit**
    - The implicit intent is the intent where instead of defining the exact components, you define the action you want to do.
    - An Implicit intent specifies an action that can invoke any app on the device to be able to perform an action. Using an Implicit Intent is useful when your app cannot perform the action but other apps probably can and you’d like the user to pick which app to use.
* Exapmle
* Intent i=new Intent();
* i.setAction(Intent.ACTION\_SEND);
  + **Explicit**
    - When you explicitly define the component that needs to be called by the Android System
    - Explicit intent is one that you can use to launch a specific app component, such as a particular activity or service in your app
* Intent I = new Intent(getApplicationContext(),NextActivity.class);
* I.putExtra(“value1” , “This value for Next Activity”);
* I.putExtra(“value2” , “This value for Next Activity”);

**onSaveInstanceState**

* If you save the state of the application in a bundle (typically non-persistent, dynamic data in onSaveInstanceState), it can be passed back to onCreate if the activity needs to be recreated (e.g., orientation change) so that you don't lose this prior information. If no data was supplied, savedInstanceState is null.

**Relative Layout.**

* Everything is positioned with relation to others.
* You can have multiple items on each other and toggle their visibility as needed

For text we tend to use the sp rather than dp for pixel size. Since sp is scale independent and in it can

**findViewById(R.id.buuton1);**

* findViewById is a method to find the view from the layout resource file that are attached with current activity
* R is class that contains all the ids of all the views.

# **Java 101 free online course**

**extends keyword**

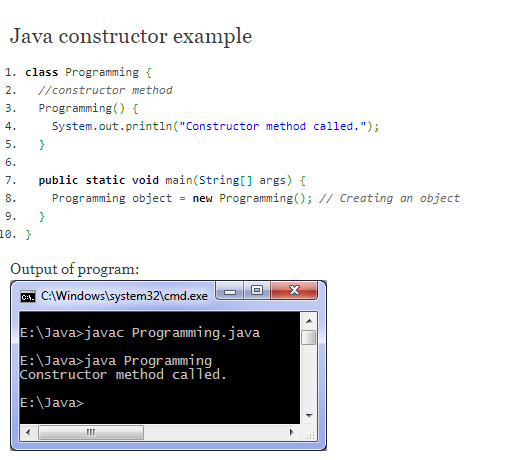
* Shows a child-parent relationship between the classes of a program.
* To avoid the class inheritance use the keyword “final”.

**Definition of a constructor**

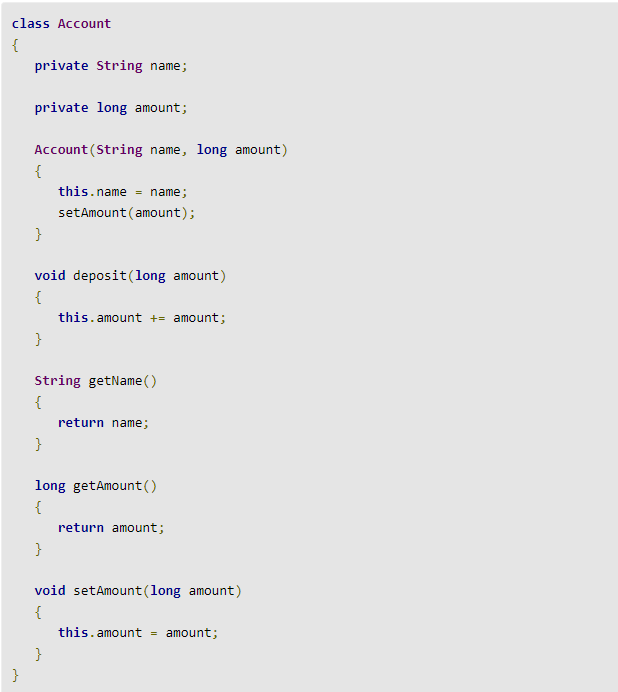
* + A block of a code that is similar to a method that’s called when an instance of an object is created.

**Differences between a method and a constructor**

* + Does not have a return type
  + The name of the constructor must be the same as the name of the class
  + Unlike methods constructors are not considered a member of a class.
  + Constructor is called automatically when a new instance of an object is created.
* From my understanding, constructor is a method that is created whether you want it or not that has the exact same name as your class or object in general, and it gets created every time an instance of the object is declared.



As you can see in the above example, the construct is outputted when the class is declared in the program



In the example above, we can see that the constructor has an initial name and calls setAmount. This means that the every time you declare a new account you need to initialize it with a name and an amount. The deposit, getAmount, and getName are methods…

# **Don’t Forget**

**When and where to call super()**

Just as this() must be the first element in a constructor that calls another constructor in the same class, super() must be the first element in a constructor that calls a constructor in its superclass. If you break this rule the compiler will report an error. The compiler will also report an error if it detects a super() call in a method; only ever call super() in a constructor.