What is the difference between a JDK and a JRE?

"JRE“ – for users. Java Runtime Environment. An implementation of the Java Virtual Machine which executes Java programs.

"JDK“- for developers. Java Development Kit, used to develop Java based software. Contains JRE(s), compiler, debuggers, dev libraries etc. You need a JDK to compile

Where are they located? (windows command line demo)

```
C:\Users\Perkins>java -version
java version "1.7.0_51"
Java(TM) SE Runtime Environment (build 1.7.0_51-b13)
Java HotSpot(TM) 64-Bit Server VM (build 24.51-b03, mixed mode)

C:\Users\Perkins>where java
C:\Windows\System32\java.exe
C:\Program Files\Java\jdk1.7.0_51\bin\java.exe
```

From http://www.java-tips.org/java-se-tips/java.lang/what-is-the-difference-between-jdk-and-jre.html

Interfaces
Used to define classes of behavior

Define a type of behavior

- Abstract (methods empty,
- derived classes fill in)
- *You cannot instantiate an interface.*
- *An interface does not contain any constructors.*
- *All of the methods in an interface are abstract.*
- *An interface cannot contain instance fields. The only fields that can appear in an interface must be declared both static and final.*
- *An interface is not extended by a class; it is implemented by a class.*

```
interface animal {
    public void eat();
    public void travel();
}

public class Mammal
implements animal{
}
```

Show how you are forced to override methods or make class abstract

Demo using Animation.AnimationListener in a class

Don’t use magic numbers

```java
@Override
public void resume() {
    //problem here, 10 is a magic number,
    //10 what? what does it mean?
    initDeals(10);
}
```

Bad

```java
//better idea, define a constant
//make it static so it is only allocated once
//not every time enclosing object allocated
//make it final so it cannot be changed
public static final int NUMBER_DEFAULT_DEALS = 10;

@Override
public void resume() {
    initDeals(NUMBER_DEFAULT_DEALS);
}
```

Good

Also note that the name is all caps, convention states that a variable in all caps is a constant

Consider defining constants in 1 place

```
public final class Constants {

      public static final boolean PASSES = true;
      public static final boolean FAILS = false;

      //static helper class do not
      //need to be constructed
      private Constants(){ }
}

//usage
boolean myGrade = Constants.PASSES;
```

See http://www.javapractices.com/topic/TopicAction.do?Id=2

If you need a chunk of code more than one time – extract a function

```java
int newFlower;
newFlower = rand.nextInt(CONSTANTS.NUMB_FLOWERS) + 1;
if (newFlower== 1)
    f1.setImageResource(R.drawable.f1);
if (newFlower== 2)
    f1.setImageResource(R.drawable.f2);
if (newFlower== 3)
    f1.setImageResource(R.drawable.f3);

newFlower = rand.nextInt(CONSTANTS.NUMB_FLOWERS) + 1;
if (newFlower== 1)
    f1.setImageResource(R.drawable.f1);
if (newFlower== 2)
    f1.setImageResource(R.drawable.f2);
if (newFlower== 3)
    f1.setImageResource(R.drawable.f3);

newFlower = rand.nextInt(CONSTANTS.NUMB_FLOWERS) + 1;
if (newFlower== 1)
    f1.setImageResource(R.drawable.f1);
if (newFlower== 2)
    f1.setImageResource(R.drawable.f2);
if (newFlower== 3)
    f1.setImageResource(R.drawable.f3);
```

```java
@Override
public void onAnimationEnd(Animation animation) {
    Log.d(TAG, "onAnimationEnd: ");

    f1_val = ChangeImage(f1);
    f2_val = ChangeImage(f2);
    f3_val = ChangeImage(f3);

    calculateScore();
}

/**...*/
private int ChangeImage(ImageView myView) {
    int newFlower = rand.nextInt(CONSTANTS.NUMB_FLOWERS) + 1;
    if (newFlower == 1)
        myView.setImageResource(R.drawable.f1);
    if (newFlower == 2)
        myView.setImageResource(R.drawable.f2);
    if (newFlower == 3)
        myView.setImageResource(R.drawable.f3);
    return newFlower;
}
```

Java is always pass-by-value. The difficult thing to understand is that Java passes objects as references and those *references* are passed by value. You can change what it points to but not original reference

Uninstall and ReInstall to test

- When you are finished with an app, completely uninstall and then reinstall it.
- Then test it.
- This should highlight any default ommisions.

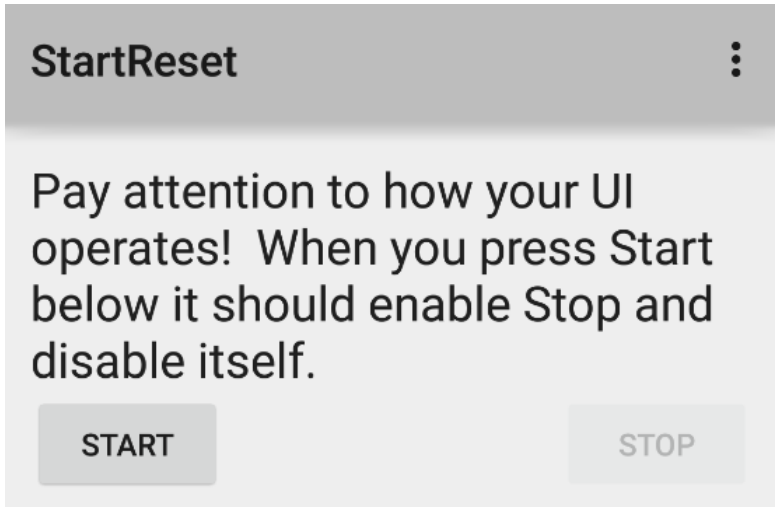
Version Control

- Local history
  - Tracks changes to your files
  - Can apply labels (like a git Tag)
  - Easy to rollback mistakes
  - Problems:
    - No remote storage (lose your laptop, lose your history)
    - File->Invalidate Caches/Restart gets rid of all local history

Version Control

- You will be using it later so might as well ease your burden and learn now
- Built in SVN handlers
- Git is the big one, Install git
- Where to store?
  - Github - get free account with free private repositories
  - BitBuccket – github clone with free repos, need plugin for AS, not sure if reliable. (see https://bitbucket.org/atlassian/jetbrains-bitbucket-connector)
- Demo on AS

Pay attention to UI function!




```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Start"
    android:id="@+id/b_start"
    android:layout_below="@+id/textView"
    android:layout_alignParentStart="true"
    android:onClick="doStart"/>

<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Stop"
    android:id="@+id/b_stop"
    android:layout_below="@+id/textView"
    android:layout_alignEnd="@+id/textView"
    android:onClick="doStop"/>
```

```
private boolean bStartEnabled=true;
Button bStart;
Button bStop;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    bStart = (Button)findViewById(R.id.b_start);
    bStop  = (Button)findViewById(R.id.b_stop);
    setButtonState(bStartEnabled);
}

private void setButtonState(boolean bStartEnabled) {
    bStart.setEnabled(bStartEnabled);
    bStop.setEnabled(!bStartEnabled);
}
public void doStart(View view) {
    bStartEnabled = false;
    setButtonState(bStartEnabled);
}

public void doStop(View view) {
    bStartEnabled = true;
    setButtonState(bStartEnabled);
}
```

So you want to make your own icons

- Be sure to make them in every size (xdpi, mdpi, ldpi), otherwise they are scaled by android
- Demo ic_menu_share in sdk\platforms\android-19\data\res
- Make sure they complement any standard android icons you use.

So you want to make your own icons

- Make sure they are transparent

- Otherwise it looks like this

Look up ‘transparent icon’ in google

So you want to make your own icons

- Or you can use standard icons for standard stuff (refresh, edit, share…)
- Look in your [SDK]/platforms/android-[VERSION]/data/res.
- In XML something like

```
android:icon="@android:drawable/ic_menu_edit"
```

- Also, do not Manually copy android icons into your /res folders

Use LogCat

```
public class HelloAndroid extends Activity {

    //used by logcat, useful for filtering in LogCat view
    private static final String TAG = "HelloAndroidActivity";


    protected void onDestroy() {
        super.onDestroy();

        //log the fact that we entered onDestroy
        Log.i(TAG,"onDestroy");
```

1. Define Tag for filtering

2. Log Message

3. Create filter

4. Debug app to View Log. Found in DDMS Or Added via Window->Show View->LogCat