CS 646 Android Mobile Application Development
Spring Semester, 2015
Doc 9 Coding Style, Dialogs, Concurrency
Feb 26, 2015

Copyright ©, All rights reserved. 2015 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/openpub/) license defines the copyright on this document.

Coding Style

Formatting

Format your code

- Uniformly
- Consistently
- Show the block structure of your code

```
public void commandAction(Command c, Displayable d) {
   if (c == restartCmd) {
theGame.restart();
   } else if (c == levelCmd) {
   Item[] levelItem = {
 new Gauge(“level”, true, 9 theGame.getLevel())};
Form f = new Form(“Change Level”, levelItem);
     f.addCommand(OkCmd);
 f.addCommand(cancelCmd);
  f.setCommandListener(this);
    Display.getDisplay(this).setCurrent(f);
    } else if (c == exitCmd) {
 destroyApp(false);
 notifyDestroyed();
  }
}
```

```
public void commandAction(Command c, Displayable d) {
if (c == restartCmd) {
theGame.restart();
} else if (c == levelCmd) {
Item[] levelItem = {
new Gauge(“level”, true, 9 theGame.getLevel())};
Form f = new Form(“Change Level”, levelItem);
f.addCommand(OkCmd);
f.addCommand(cancelCmd);
f.setCommandListener(this);
Display.getDisplay(this).setCurrent(f);
} else if (c == exitCmd) {
destroyApp(false);
notifyDestroyed();
}
}
```

Name Structure - Language Conventions

	Java
Class	PascalCase
Method	camelCase
Field	camelCase
Parameter	camelCase
Local Variable	camelCase

PascalCase ArrayList

camelCase courseSize

Reading Verses Writing Programs

Code

- Written once
- Read many times

Use names that help the reader understand the code

Avd brvtns

brvtns r hrd t rd

n brvtn cn stnd fr dffrnt thngs

tmp - tmprr r tmprtr

Dffrnt ppl wll brvt dffrntl

Ds tcmplt s dn't hv t typ lng nms

Avoid Abbreviations

Abbreviations are hard to read

An abbreviation can stand for different things

tmp - temporary or temperature

Different people will abbreviate differently

IDEs autocomplete so don't have to type long names

Describe What "flag" is Used For

```
if (flag) {
        ...
}
```

```
if (foundDuplicate) {
        ...
}
```

flag

flagStatus

computeFlag

Do not help understand code

Variables 1 through N

String s1;
String s2;

String fileContents;
String pattern;

Who can remember the difference between s1 and s2?

Avoid Names With No Meaning

MyLinkedList

Who are you?
What makes your LinkedList different?

temp

All variables are temporary

```
swap = a;
a = b;
b = swap
```

Don’t Just Repeat the Type


TextView tv;


TextView textView;


TextView userName;


TextView methodsCalled;

Guidelines - Class Names

Use nouns

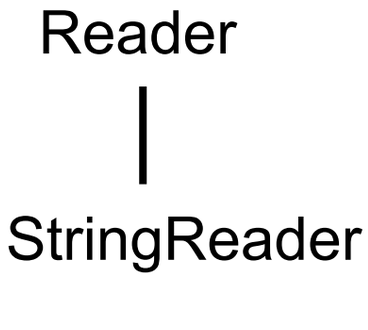
No abbreviations

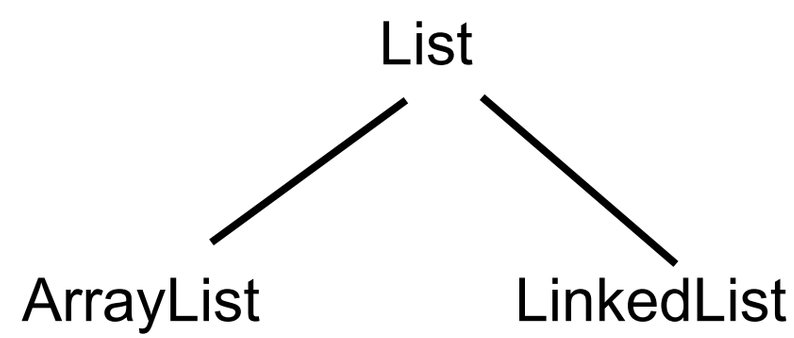
Superclass

Single word to convey its purpose

Subclass

Prepend adjective to superclass name





Guidelines - Method/Function/Procedure Names

Describe what method does

Use verb to describe an action

add(int index, E element)
clear()

If returns a value name what it returns

iterator()
subList(int fromIndex, int toIndex)

If returns boolean value make it true/false statement

isEmpty()
contains(Object o)

Guidelines - Variables, Fields, Parameters

Use names that indicate role variable is playing
If declare variable types don't use type as name
Use plurals to indicate collections
Make boolean variable names true/false statement
isVisible, hasMultipleParts,

```
public void execute(Vector vector) {
        Stack s;
}
```

```
public void execute(Vector commands) {
        Stack commandsExecuted;
}
```

Summary

Use names to help the reader understand the code

Follow language conventions

Avoid abbreviations

Use names that indicate role item is playing

Dialogs

Types of Dialogs

AlertDialog

Can have buttons and checkboxes

ProgressDialog

DatePickerDialog

TimePickerDialog

Custom Dialogs

See http://developer.android.com/guide/topics/ui/dialogs.html

Activity.onCreateDialog(int)

```
static final int DIALOG_PAUSED_ID = 0;
static final int DIALOG_GAMEOVER_ID = 1;
```

Create dialogs in onCreateDialog

```
protected Dialog onCreateDialog(int id) {
    Dialog dialog;
    switch(id) {
    case DIALOG_PAUSED_ID:
        // do the work to define the pause Dialog
        break;
    case DIALOG_GAMEOVER_ID:
        // do the work to define the game over Dialog
        break;
    default:
        dialog = null;
    }
    return dialog;
}
```

showDialog(int)

To show a dialog in your activity call showDialog(int) which calls onCreateDialog the first time

```
showDialog(DIALOG_PAUSED_ID);
```

Creating an AlertDialog

Class DialogExample

```
protected Dialog onCreateDialog(int id) {
        switch (id) {
        case SAMPLE_DIALOG_ID:
                AlertDialog.Builder builder = new AlertDialog.Builder(this);
                builder.setTitle("Hello").setPositiveButton("Ok",
                                new DialogInterface.OnClickListener() {
                                        public void onClick(DialogInterface dialog,
                                                        int whichButton) {
                                                DialogExample.this.finish();
                                                Toast.makeText(getApplicationContext(), "Good Bye",
 Toast.LENGTH_SHORT).show();
                                        }
                                });
                        return builder.create();
                default:
                        return null;
                }
        }
```

Three Buttons

Positive

Negative

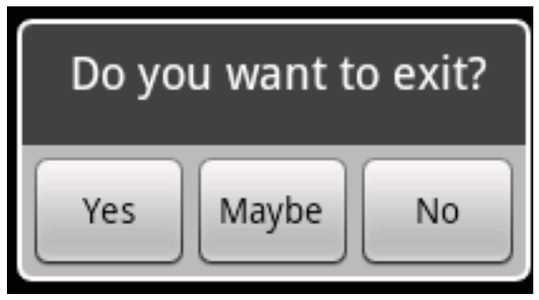
Neutral

Can have only one of each

Button types have no meaing

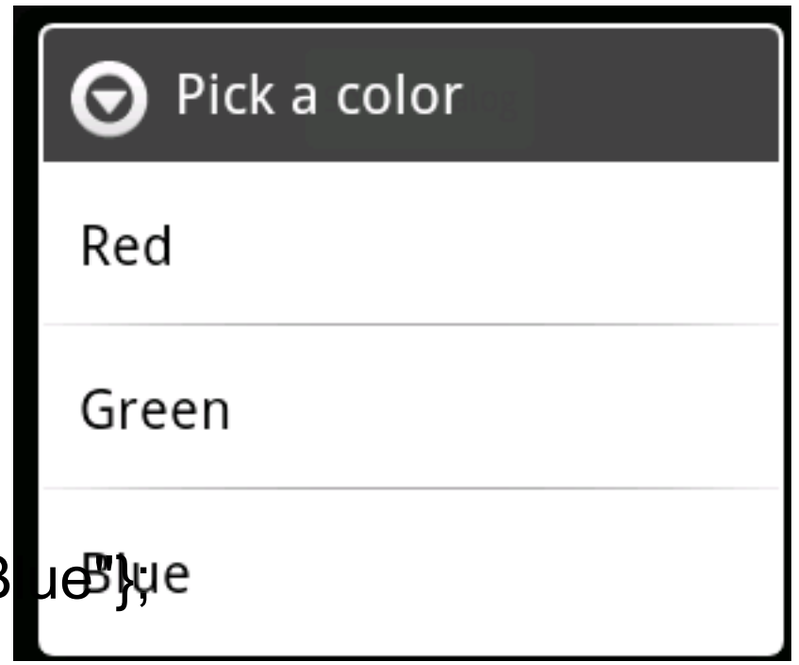
Positive can do what every you want

Three Button Example




```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("Do you want to exit?")
        .setCancelable(false)
        .setPositiveButton("Yes",
                new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
                                        int whichButton) {
                                Toast.makeText(getApplicationContext(), "Good Bye",
                                                Toast.LENGTH_SHORT).show();
                                DialogExample.this.finish();
                        }
                })
                .setNegativeButton("No",
                new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
                                        int whichButton) {
                                dialog.cancel();
                        }
                })
                .setNeutralButton("Maybe",
                new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog,
                                        int whichButton) {
                                Toast.makeText(getApplicationContext(), "Make up your mind",
                                                Toast.LENGTH_SHORT).show();
                                DialogExample.this.showDialog(SAMPLE_DIALOG_ID); //Does not work
                        }
                });
    return builder.create();
```

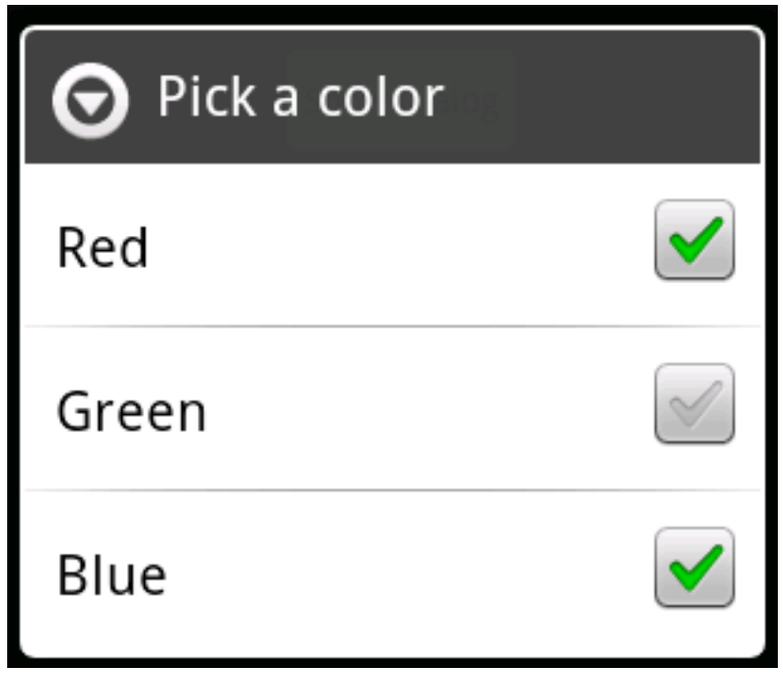
Lists




```
protected Dialog onCreateDialog(int id) {
    switch (id) {
    case SAMPLE_DIALOG_ID:
        final CharSequence[] items = {"Red", "Green", "Blue"};

        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Pick a color");
        builder.setItems(items, new DialogInterface.OnClickListener() {
            public void onClick(DialogInterface dialog, int item) {
                Toast.makeText(getApplicationContext(), items[item],
                Toast.LENGTH_SHORT).show();
            }
        });
        return builder.create();
    default:
        return null;
    }
}
```

MultiSelection




```
protected Dialog onCreateDialog(int id) {
        switch (id) {
        case SAMPLE_DIALOG_ID:
                final CharSequence[] items = {"Red", "Green", "Blue"};
                final boolean[] selected = {false, true, false};

                AlertDialog.Builder builder = new AlertDialog.Builder(this);
                builder.setTitle("Pick a color")
                        .setMultiChoiceItems(items, selected, new DialogInterface.OnMultiChoiceClickListener() {
                    public void onClick(DialogInterface dialog, int item, boolean isChecked) {
                        Toast.makeText(getApplicationContext(), items[item] + " isChecked " + isChecked,
                     Toast.LENGTH_SHORT).show();
                        }
                    });
                return builder.create();
        default:
                return null;
        }
}
```


Thursday, February 26, 15

Should add a button to let the user exit. Also need keep track of which items are selected. Docs say that the Dialog has that information - dialog.getListView().isItemChecked(int position) or dialog.getListView().getCheckedItemPositions()

Concurrency in Android

Processes and Threads

Processes have own address space

- Take longer to start
- Consume more memory

Threads share address space

Android, Processes and Threads

Android application starts with

- One process running one thread
- The thread is called the main or UI thread
- Activity code runs in main (UI) thread

Can create more threads to run in same process

Can configure activities to run in separate processes

- Not as common as creating threads

Android Thread Rules

Don't block the UI thread

- Activity code runs on the UI thread
- Create threads to perform long operations

Do not access the Android UI toolkit from outside the UI thread

- Use the following to access UI thread
  - Activity.runOnUiThread(Runnable)
  - View.post(Runnable)
  - View.postDelayed(Runnable, long)

Actually these are the rules for all UI frameworks that I have used

Runnable

Java Interface

void run()

Android Background Tools

Java threads

Handler

- Messages
- Runnables

AsyncTask

Services

AsyncTask

Why AsyncTask

Make it easier to deal with threads

Handle the common case for you

AsyncTask

Replaces threads & Messages

Android 1.5

Subclass AsyncTask

onPreExecute()

Run in UI thread

Done first

doInBackground(Params...)

Run in seperate thread

publishProgress(Progress...)

Call in doInBackground() to register progress

onProgressUpdate(Progress...)

Run in UI thread

Called by publishProgress

onPostExecute(Result)

Run in UI thread

Run after doInBackground ends

Rules

The AsyncTask subclass instance must be created on the UI thread

execute(Params...)

Starts the task

Must be invoked on the UI thread

Do not call manually

onPreExecute(), onPostExecute(Result), doInBackground(Params...), onProgressUpdate(Progress...)

The task can be executed only once

AsyncTask Types

private class SampleTask extends AsyncTask<Params, Progress, Result>

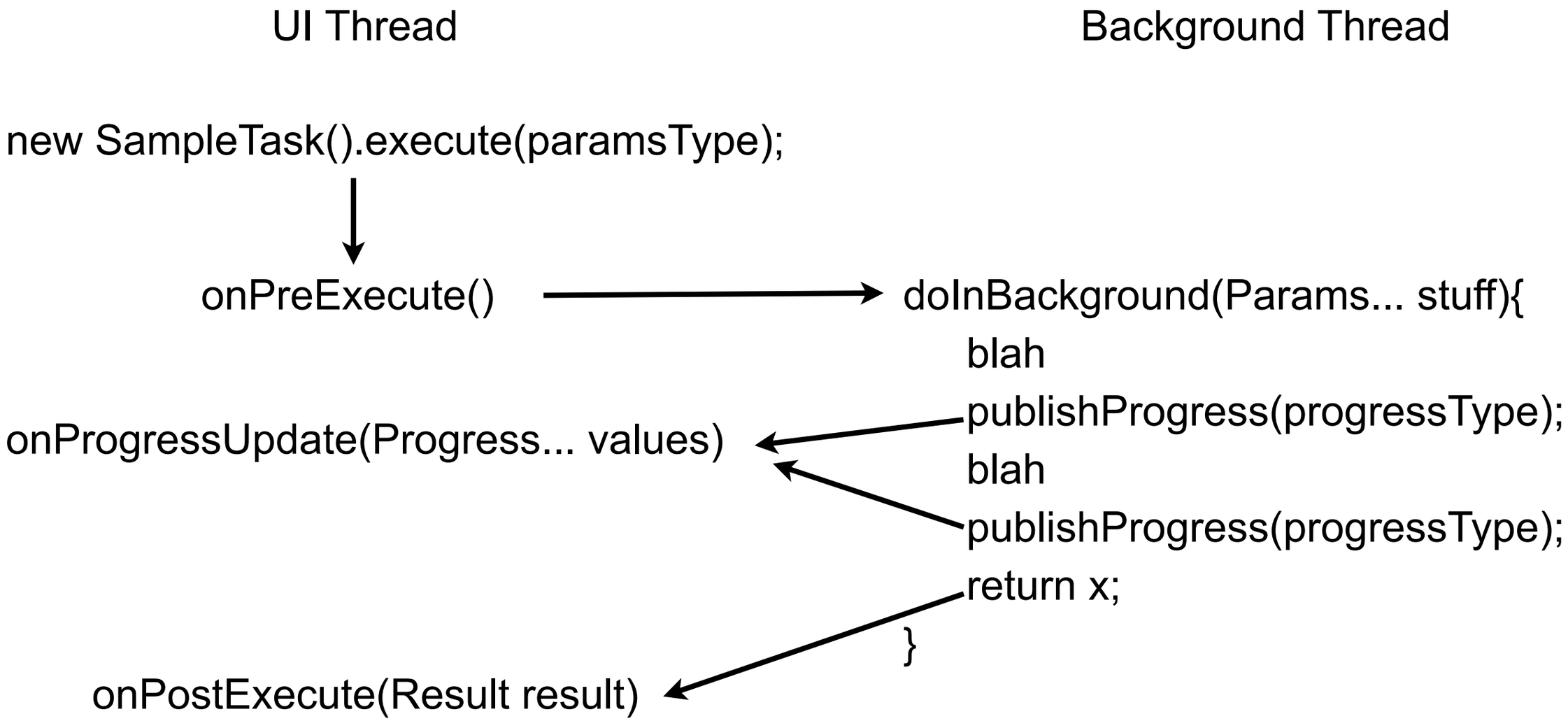
Params
- Type of argument for doInBackground()
- execute()

Progress
- Type of argument for publishProgress()
- onProgressUpdate()

Result
- Return type for doInBackground()
- Type of argument for onPostExecute()

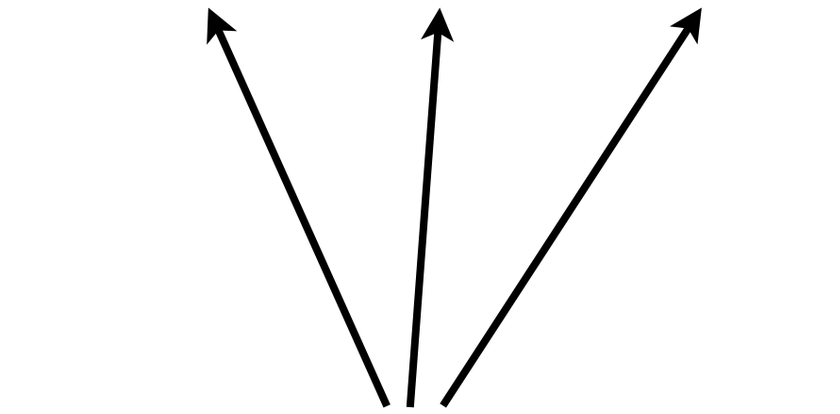
How it Works

class SampleTask extends AsyncTask<Params, Progress, Result>



Generics

class SampleTask extends AsyncTask<Params, Progress, Result>



Need to know the actual values

Example

Loops in the background and displays Toast

ThreadExample

```
public class ThreadExample extends Activity {

    private class SampleTask extends AsyncTask<String, String, Void> {
        protected Void doInBackground(String... words) {
            for (String word : words) {
                publishProgress(word);
                SystemClock.sleep(1000);
            }
            return (null);
        }

        protected void onPostExecute(Void unused) {
            Toast.makeText(ThreadExample.this, "Done", Toast.LENGTH_SHORT)
                    .show();
        }
```

ThreadExample

```
        protected void onPreExecute() {
            Toast.makeText(ThreadExample.this, "Start", Toast.LENGTH_SHORT).show();
        }

        protected void onProgressUpdate(String... word) {
            Toast.makeText(ThreadExample.this, word[0], Toast.LENGTH_SHORT).show();
        }
    }

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onStart() {
        super.onStart();
        String[] text = { "Bat", "cat", "dat", "fat", "hat", "mat" };
        new SampleTask().execute(text);
    }
}
```

Handler

Handler

Used to:

Send message from one thread to another

Execute code in the future

Uses Message object

android.os.Message

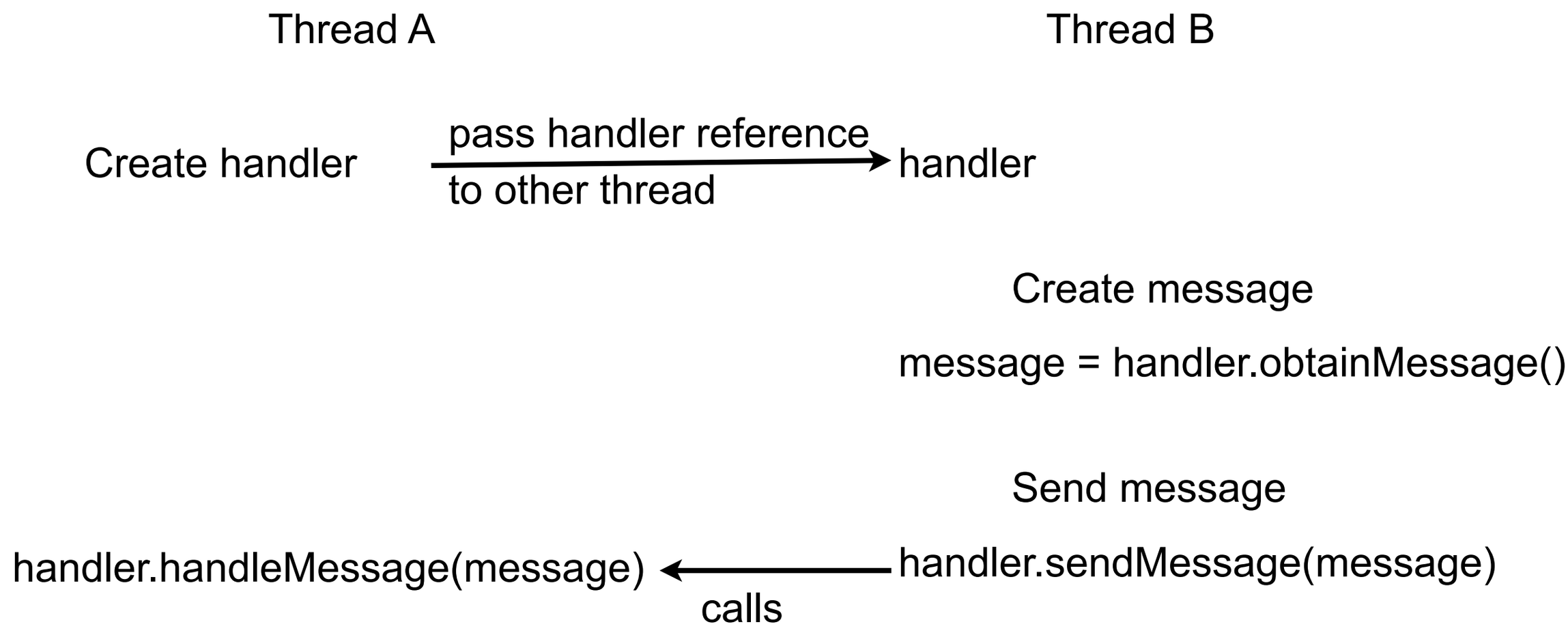
Contains public fields for data

- int arg1
- int arg2
- Object obj
- int what

Data fields mean what ever you want

Can add a bundle for more data

How it Works


Thread A
Thread B
Create handler
pass handler reference
to other thread
handler
Create message
message = handler.obtainMessage()
Send message
handler.handleMessage(message)
calls
handler.sendMessage(message)

Creating Messages - Handler methods

obtainMessage()

obtainMessage(int what)

obtainMessage(int what, Object obj)

obtainMessage(int what, int arg1, int arg2)

obtainMessage(int what, int arg1, int arg2, Object obj)

Handling Messages

Handler subclass must implement

handleMessage(Message aMessage)

This method has to handle messages sent

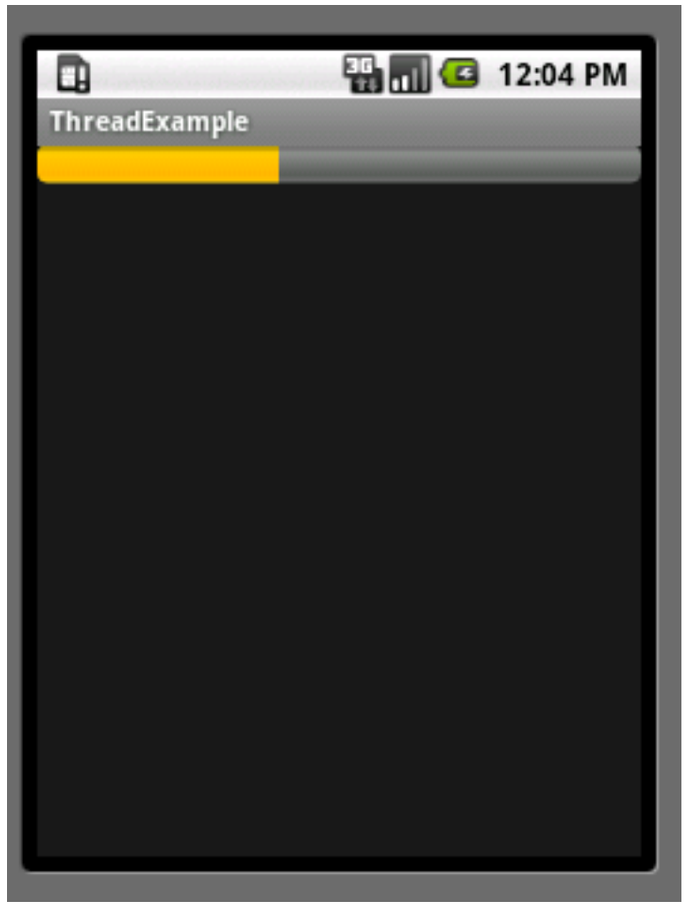
Handler Scheduling

post(Runnable)
postAtTime(Runnable, long)
postDelayed(Runnable, long)

sendEmptyMessage(int)
sendMessage(Message)
sendMessageAtTime(Message, long)
sendMessageDelayed(Message, long)

ProgressBar Example

Just shows a progress bar progressing



ThreadExample

```
public class ThreadExample extends Activity {
    ProgressBar progressView;
    boolean isRunning = false;

    Handler handler = new Handler() {
        public void handleMessage(Message empty) {
            progressView.incrementProgressBy(5);
        }
    };

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        progressView = (ProgressBar) findViewById(R.id.progress);
    }

    public void onStop() {
        super.onStop();
        isRunning = false;
    }
```

ThreadExample

```
public void onStart() {
        super.onStart();
        progressView.setProgress(0);

        Thread background = new Thread(new Runnable() {
                public void run() {
                        try {
                                for (int i = 0; i < 20 && isRunning; i++) {
                                        Thread.sleep(1000);
                                        handler.sendMessage(handler.obtainMessage());
                                }
                        } catch (Throwable t) {// just end    }
                }
        });
        isRunning = true;
        background.start();
    }


}
```

Sending Text Messages to the future

Rather than use a thread use sendMessageDelayed

Sends data in the message using Bundle

Sending Text The Hard Way

```
public class ThreadExample extends Activity {
        Handler handler = new Handler() {
                public void handleMessage(Message word) {
                        String text = word.getData().getString("key");
                        Toast.makeText(ThreadExample.this, text, Toast.LENGTH_SHORT).show();
                }
        };

        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
        }

        public void onStart() {
                super.onStart();
                String[] text = { "Bat", "cat", "dat", "fat", "hat", "mat" };
                for (int i = 0; i < text.length; i++) {
                        Bundle data = new Bundle();
                        data.putString("key", text[i]);
                        Message word = new Message();
                        word.setData(data);
                        handler.sendMessageDelayed(word, 1000 * (i + 1));
                }
```


Read the documentation to find the easy way

Progress Dialog

Displays a Progress Dialog
Uses Message what to transmit data

ThreadExample

```
public class ThreadExample extends Activity {
	ProgressDialog waitDialog;
	private static final int WAIT_DIALOG_KEY = 0;
	Handler handler = new Handler() {
		public void handleMessage(Message command) {
			if (command.what == 0)
				showDialog(WAIT_DIALOG_KEY);
			else
				waitDialog.dismiss();
		}
	};

	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
	}
```

ThreadExample

```
protected Dialog onCreateDialog(int id) {
        switch (id) {
        case WAIT_DIALOG_KEY: {
                waitDialog = new ProgressDialog(this);
                waitDialog.setMessage("Please wait while loading...");
                waitDialog.setIndeterminate(true);
                waitDialog.setCancelable(true);
                return waitDialog;
        }
        }
        return null;
}

        public void onStart() {
                super.onStart();
                Message on = new Message();
                on.what = 0;
                handler.sendMessageDelayed(on, 1000);
                Message off = new Message();
                off.what = 1;
                handler.sendMessageDelayed(off, 8000);
        }
}
```