



JÖNKÖPING UNIVERSITY

School of Engineering

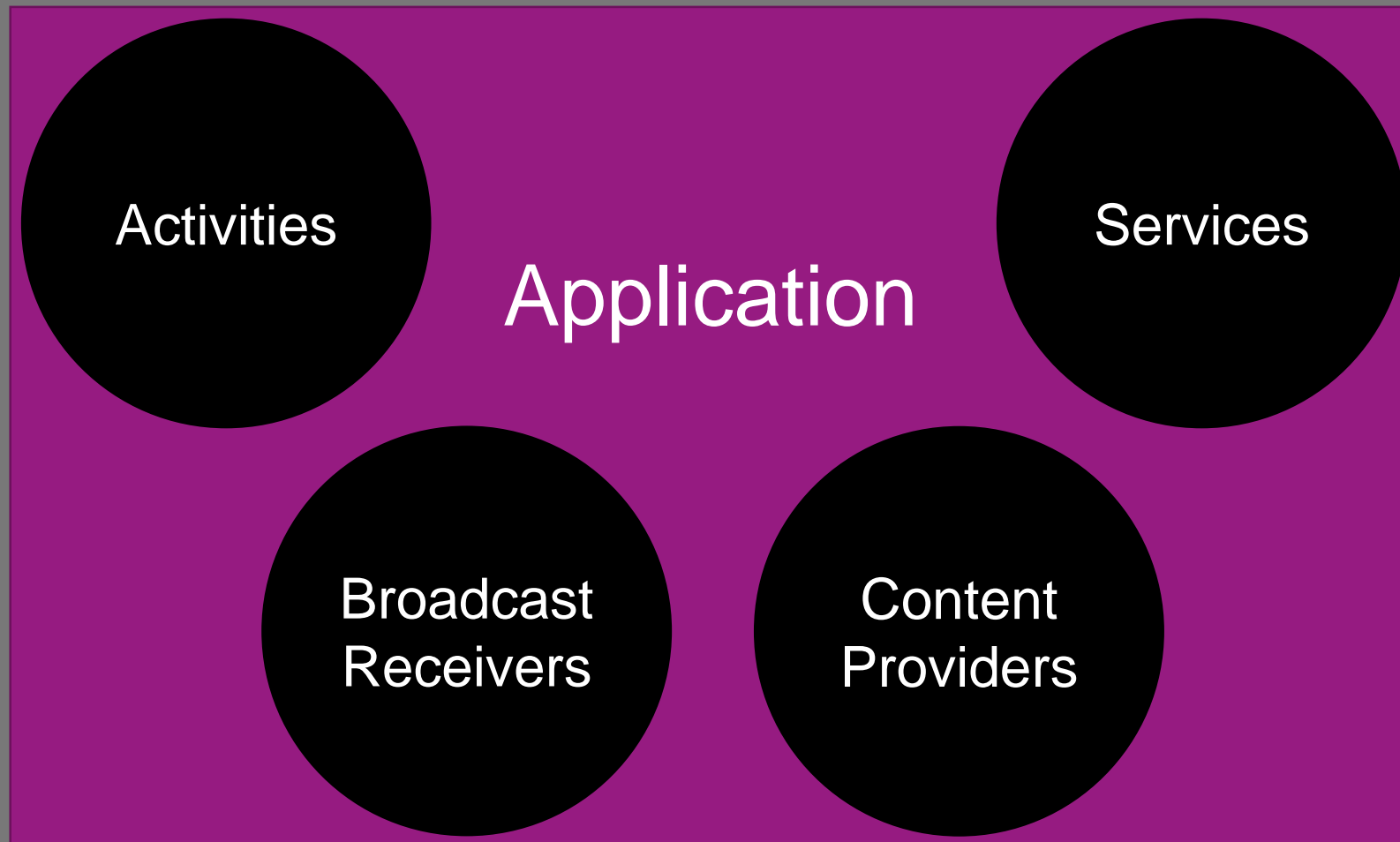
ANDROID SERVICES

Peter Larsson-Green

Jönköping University

Spring 2020

FUNDAMENTAL APP COMPONENTS



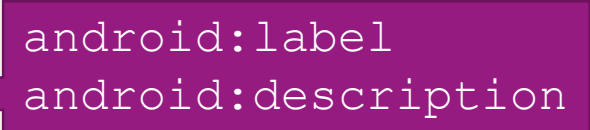
SERVICES VS THREADS

- Threads in activities should not outlive activities.
 - Android sees components running, not threads.
- If the process is killed, the service can be restarted.
- Services run on the main application thread.

CREATING SERVICES

```
public class MyService extends Service{  
    // Override methods to implement your specific behavior.  
}
```

```
<manifest  
    xmlns:android="http://schemas.android.com/apk/res/android"  
    package="the.package">  
    <application ...>  
        <service android:name="the.package.TheClass" />  
    </application>  
</manifest>
```



android:label
android:description

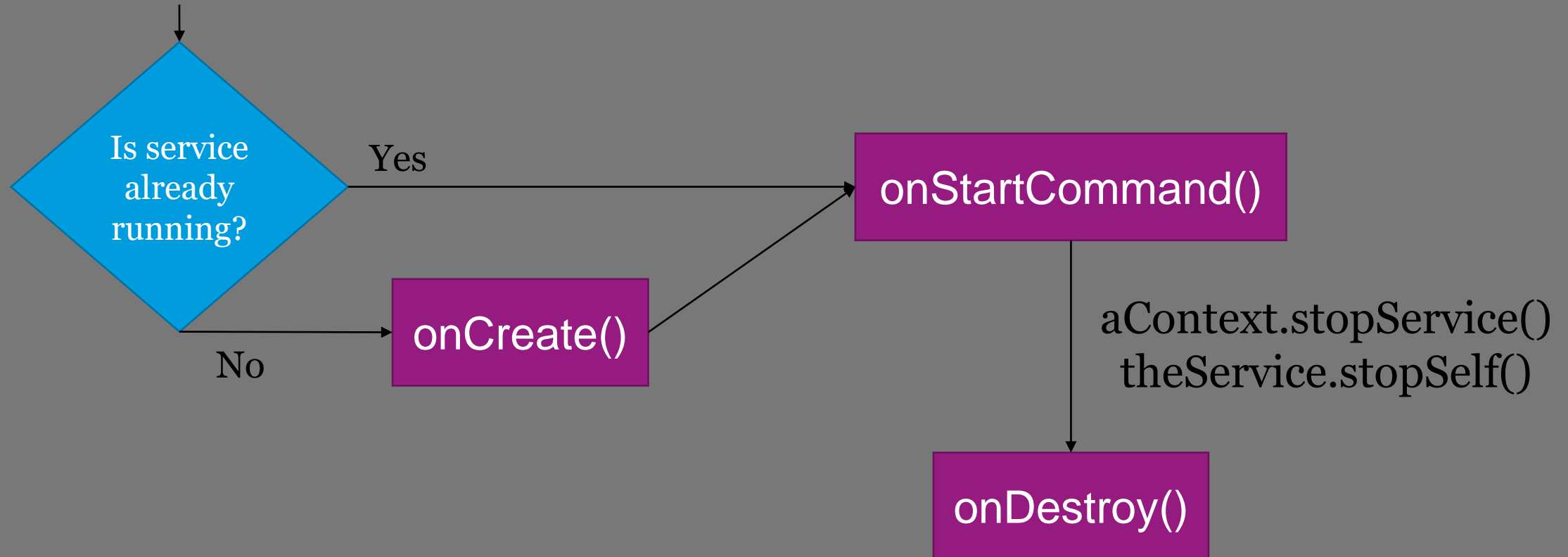
SERVICES

Services can be used in two different ways:

- As a command executor:
 1. Send a command to the service.
 2. The service executes the command.
- As a bound service:
 1. Send command to the service through an interface you define.
 2. The service can send back results to the client.

SERVICES' LIFE CYCLE (COMMAND EXECUTOR)

`aContext.startService()`



CREATING SERVICES

The intent causing this service to run.

A unique id for each call.

```
public class MyService extends Service{
    @Override
    public int onStartCommand(Intent intent, int flags,
                               int startId) {
        // Do your work!
        return Service.START_REDELIVER_INTENT;
    }
}
```

START_REDELIVER_INTENT
START_STICKY
START_NOT_STICKY

0
START_FLAG_REDELIVERY
START_FLAG_RETRY

USING A SERVICE

Starting a service:

```
Intent intent = new Intent(aContext, TheService.class);  
aContext.startService(intent);
```

Stopping a service:

```
theService.stopSelf();
```

```
theService.stopSelfResult(theIdPassedToOnStartCommand);
```

```
aContext.stopService(intentSentToStartService);
```

INTENTSERVICE

A service with a worker thread.

```
public class MyIntentService extends IntentService{  
    public MyIntentService () {  
        super ("NameOfThread");  
    }  
    @Override  
    protected void onHandleIntent (Intent intent) {  
        // Called by the worker thread.  
    }  
}
```

RUNNING IN THE FOREGROUND

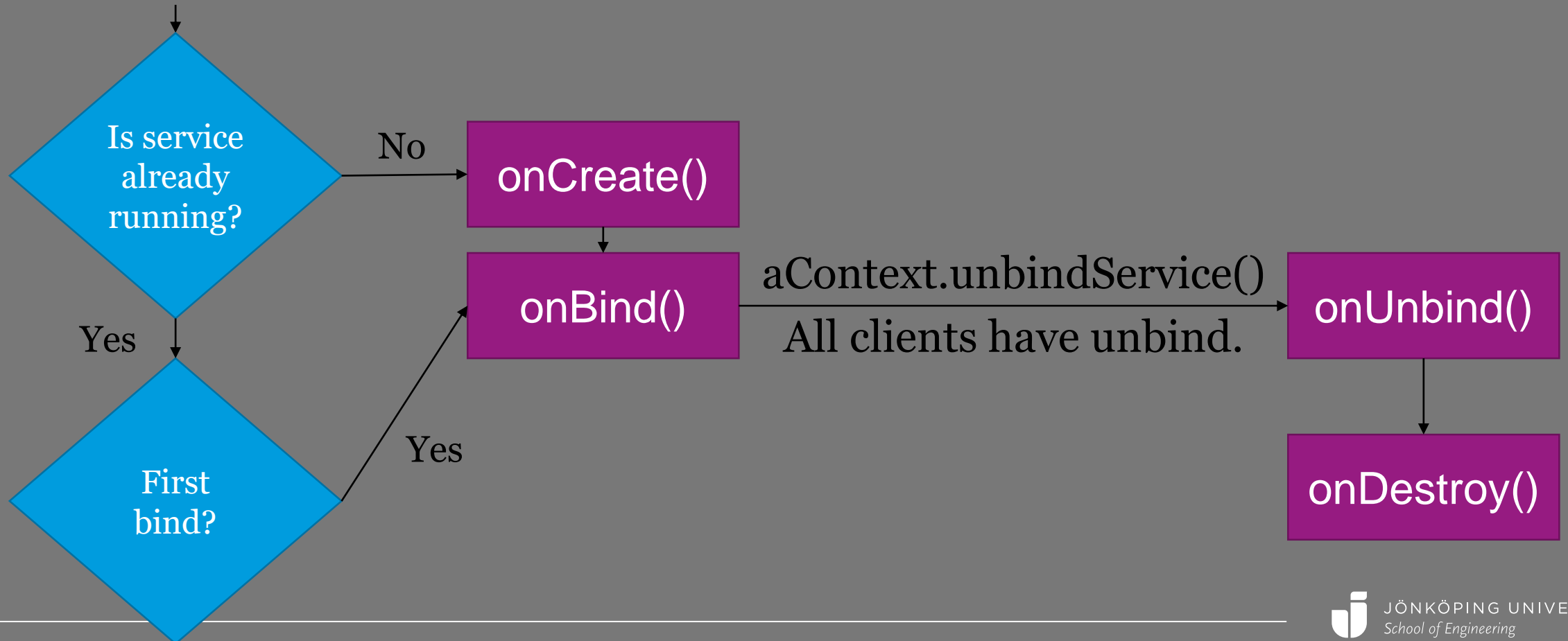
- Why?
 - To tell Android that the user is using you application.
- Consequences:
 - Your application is less likely killed.
 - Your application shows an icon in the notification bar.
- How?

```
theService.startForeground(  
    theNotificationId,  
    theNotification  
);
```

```
theService.stopForeground(bool removeNotification);
```

SERVICES' LIFE CYCLE (BOUND SERVICE)

`aContext.bindService()`



CREATING BOUND SERVICES

Act as servers (other components are clients).

```
public class MyBoundService extends Service{  
    @Override  
    public IBinder onBind(Intent intent){  
        // Return an IBinder to the client.  
    }  
}
```

CREATING BOUND SERVICES

```
public class MyBoundService extends Service{
    public class MyBinder extends Binder{
        public String getData(){
            return "This is the data!";
        }
    }
    @Override
    public IBinder onBind(Intent intent){
        return new MyBinder();
    }
}
```

USING BOUND SERVICES

```
public class MyActivity extends Activity{
    protected MyBinder myBinder;
    protected ServiceConnection connection = new ServiceConnection() {
        public void onServiceConnected(ComponentName name, IBinder service) {
            myBinder = (MyBinder) service;
        }
        public void onServiceDisconnected(ComponentName name) {
            myBinder = null;
        }
    };
}
```

USING BOUND SERVICES

```
public class MyActivity extends Activity{  
    // ...  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        Intent intent = new Intent(this, MyBoundService.class);  
        bindService(intent, connection, Context.BIND_AUTO_CREATE);  
    }  
}
```



BIND_AUTO_CREATE
BIND_ABOVE_CLIENT
BIND_NOT_FOREGROUND

USING BOUND SERVICES

```
public class MyActivity extends Activity{  
    // ...  
    protected void onDestroy() {  
        super.onDestroy();  
        unbindService(connection);  
    }  
}
```

SERVICES' LIFE CYCLE

