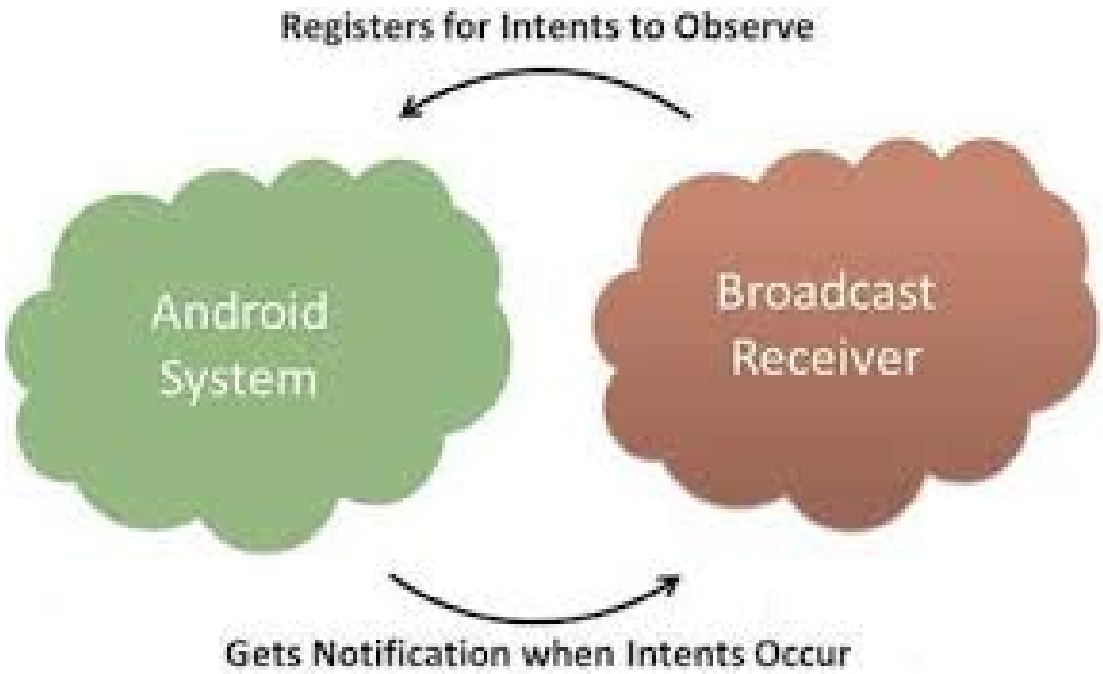
Broadcast Receiver et Intent Filters

Récepteur de diffusion et Filtres d'intention



from https://www.tutorialspoint.com

Android et les communications

- philosophie d'Android : développement d'applications riches
- pas besoin de tout réinventer + l'application doit être en mesure de bénéficier de tous les périphériques présents sur le matériel : GPS, téléphonie, réseaux, capteurs, etc.

  $\Rightarrow$ les apps doivent être modulaires et doivent être capables de tirer parti de l'ensemble des capacités des matériels $\Rightarrow$ nécessaire de mettre en place un système de communication et d'alerte au niveau du système global

  $\Longrightarrow$ modèle de communication basé sur les Intent et les Broadcast Receiver

Retour rapide sur les Intents

Communication Model

- votre application peut demander à un autre composant d'effectuer une certaine action en lui envoyant des données par l'intermédiaire d'un intent
- cette demande peut cibler uncomposant spécifique
- ou peut simplement demander l'exécution d'une action

Retour rapide sur les Intents

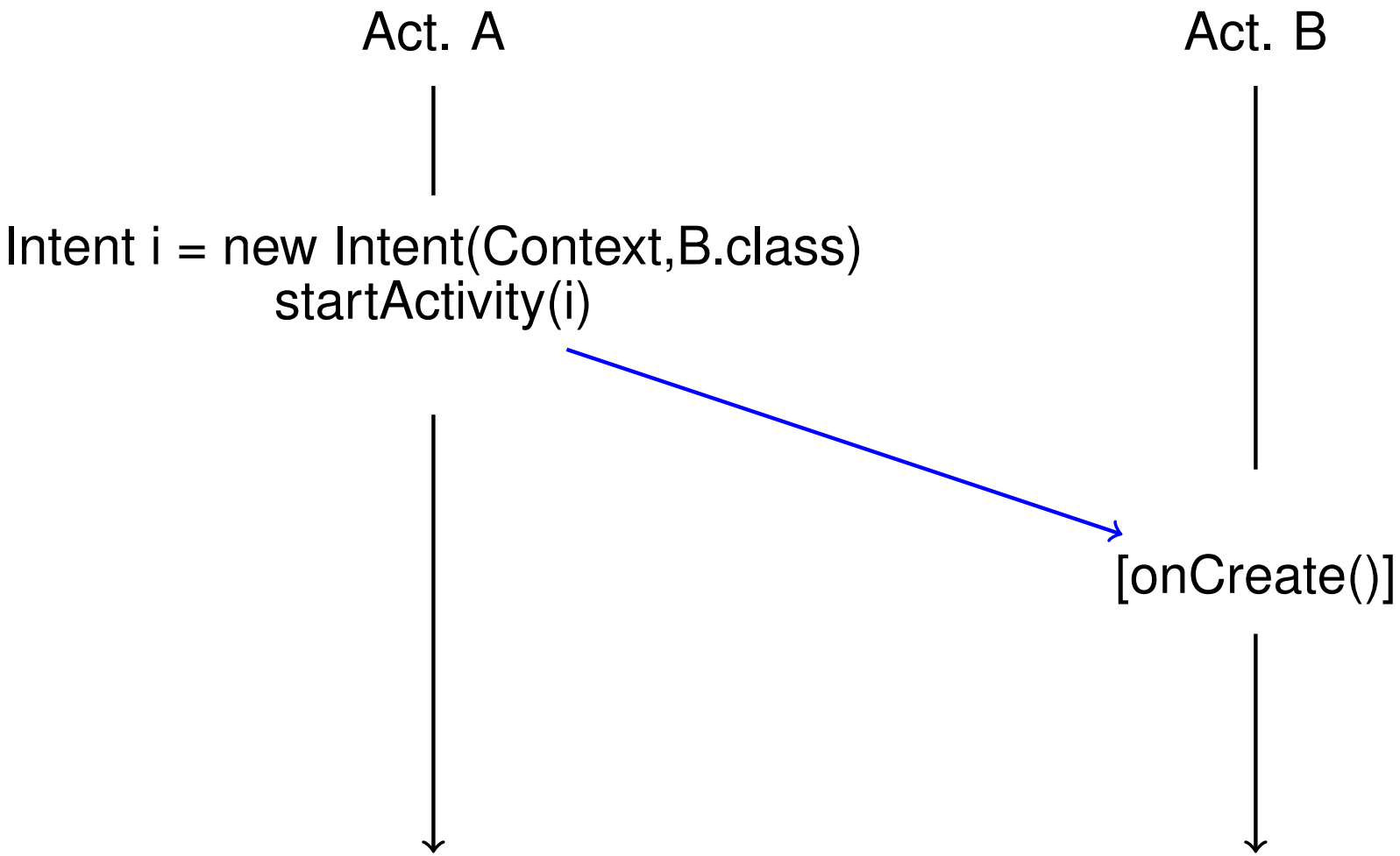
Communication Model

- votre application peut demander à un autre composant d'effectuer une certaine action en lui envoyant des données par l'intermédiaire d'un intent
- cette demande peut cibler uncomposant spécifique

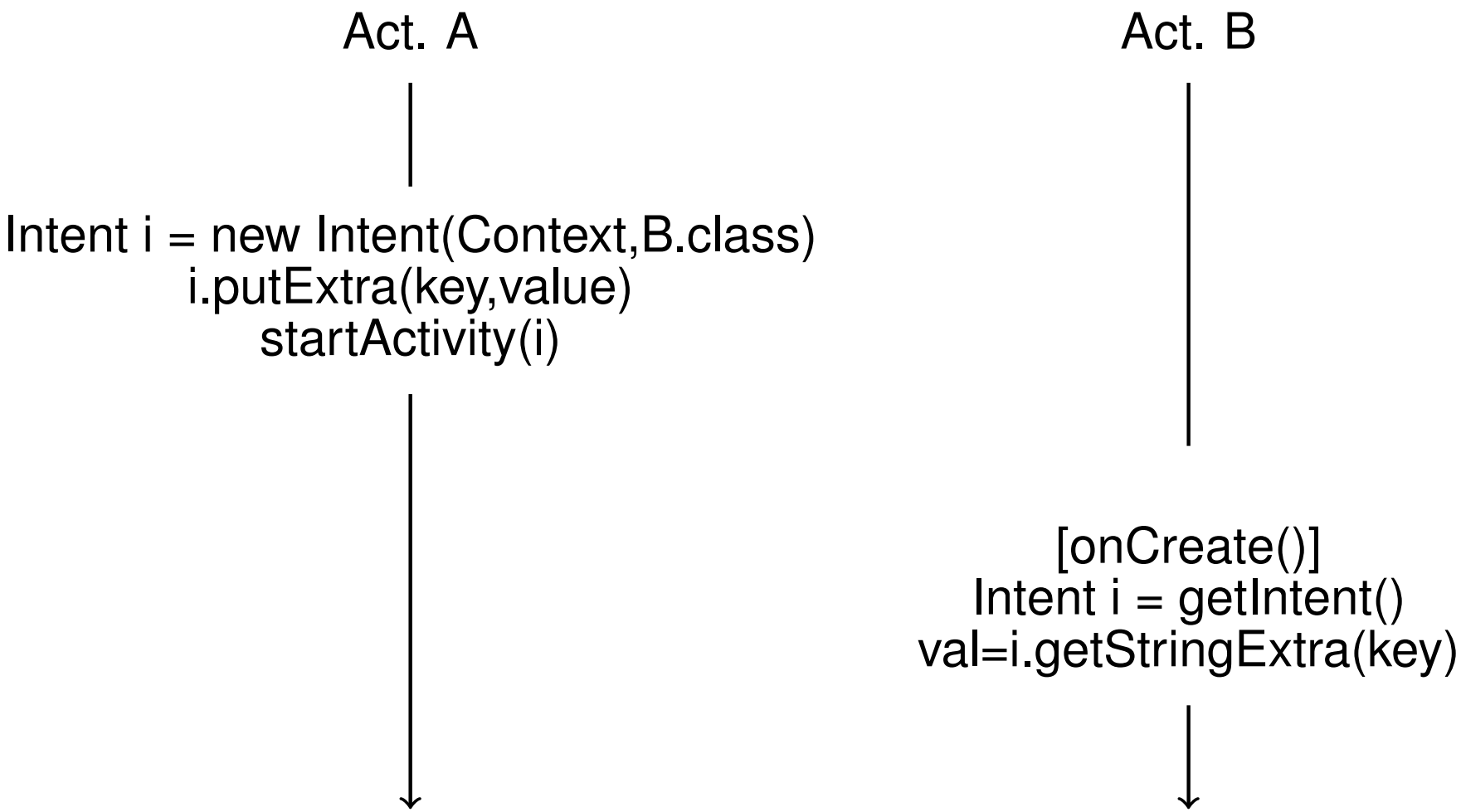
  Intent explicite
- ou peut simplement demander l'exécution d'une action

  Intent implicite

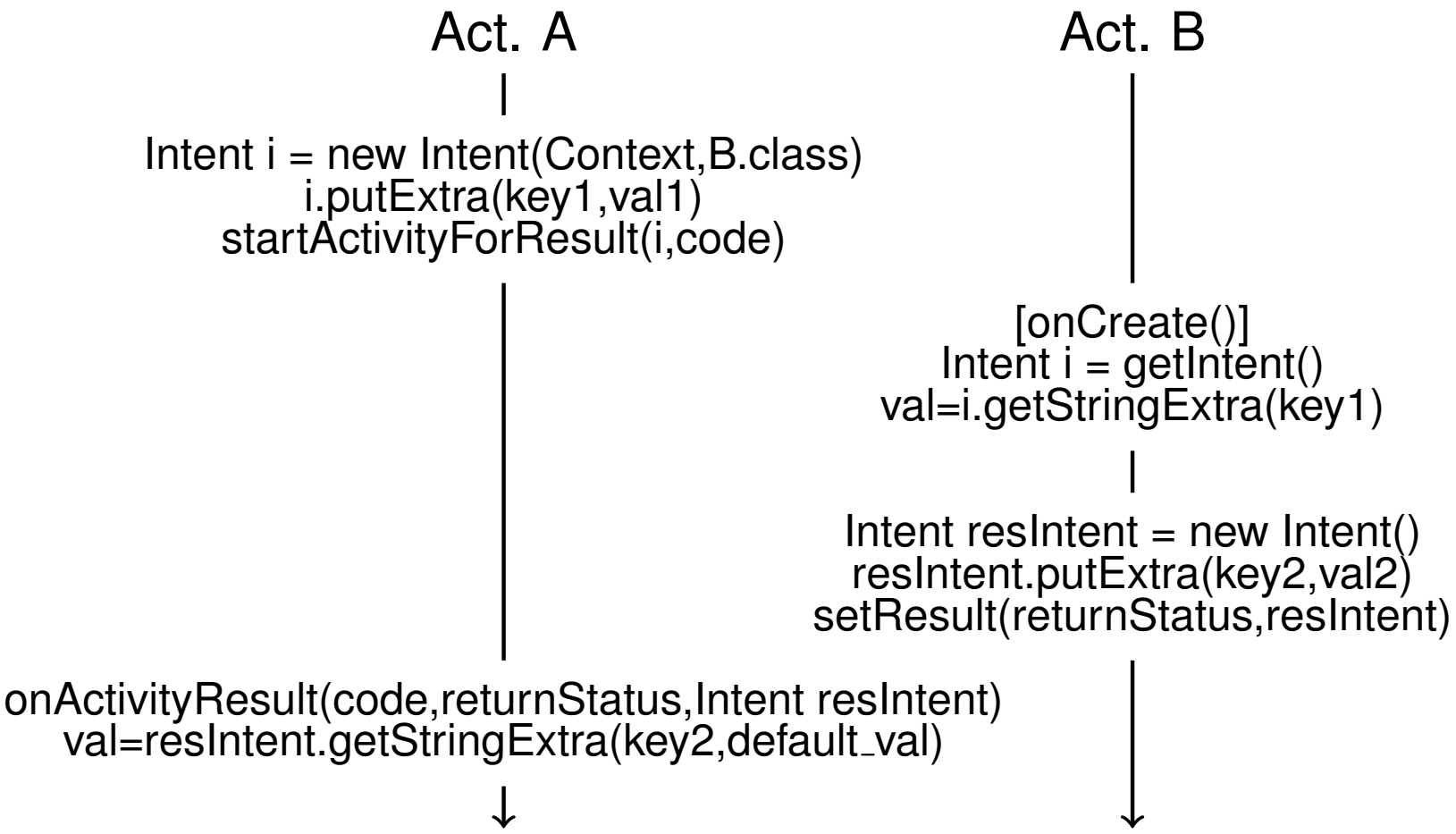
Intent explicite sans données et sans retour



Intent explicite avec données et sans retour



Intent explicite avec données et retour de résultat



Types d'intents

- Intent explicite pour lancer une app/activity dont on connait le nom de la classe
- Intent implicite pour lancer une app/activity capable d'exécuter une action particulière sur des données spécifique
- **Intents de Broadcast** qui sont utilisées par le système et par les applications pour diffuser une information au niveau global (*system-wide*)

Broadcast Intent

- de tels intents sont générés par le système pour avertir toutes les applications intéressées d'un événement particulier : e.g., GPS/Wifi s'allument ou s'éteignent, arrivée d'un nouveau sms, etc.
- votre composant peut générer de tels Intents pour avertir une autre application ou un service de la survenue d'un événement particulier pertinent dans le cadre de votre application (fin de téléchargement, sauvegarde terminée, fermeture d'une connexion réseau, etc.)
- ces broadcast intent nécessitent un objet spécifique pour les recevoir : un **Broadcast Receiver**

Broadcast Receivers

- un BroadcastReceiver est un composant Android (aux côtés des Activity, Service et ContentProvider)
- un BroadcastReceiver reçoit les intents de broadcast pour lesquels il est enregistré

  $\Longrightarrow$ si votre app doit réagir à l'arrivée d'un nouveau message, à un changement de l'état du GPS, à la connexion de votre matériel au réseau, il doit inclure un BroadcastReceiver

Tâches principales d'un BR

1. Broadcast Receivers attendent la survenue d'événements pour lesquels ils sont enregistrés
2. Broadcast Receivers reçoivent les événements
3. Broadcast Receivers réagissent à ces événements

Broadcast Receiver

Questions

- comment créer un BR ?
- comment un BR est-il connu du système ?
- comment les BR sont-ils notifiés de la survenue d'un événement particulier ?
- comment les BR prennent en charge les événements reçus ?
- comment les événements sont-ils diffusés aux BR ?

Broadcast Receiver

création

- la manière la plus simple pour créer un Broadcast Receiver est de créer une nouvelle classe qui étend la classe abstraite BroadcastReceiver et...
- d'implémenter la méthode onReceive().

```
public class MyBR extends BroadcastReceiver {
        public void onReceive(Context, Intent) {
                ...
        }
}
```

Broadcast Receiver

Comment un BR est-il connu du système ?

- pour être connu du système un BR doit simplement s'enregistrer
- un BR s'enregistre pour recevoir les événements auxquels il s'intéresse

Comment un BR s'enregistre pour un type d'action ?

- le choix de l'action se fait par l'intermédiaire d'un intent-filter (filtre d'intention).
- l'enregistrement peut être effectuée soit de manière statique par une modification du fichier AndroidManifest.xml
- ou dynamiquement en spécifiant les intent-filters dans le code

Broadcast Receiver

Enregistrement statique

- insertion dans le fichier AndroidManifest.xml à l'intérieur de la balise <application>...</application>
  - une balise <receiver>...< /receiver> avec entre ces deux balises
  - une ou plusieurs balises <intent-filter>...< /intent-filter>
- la balise `receiver` a plusieurs attributs :
  - le nom de la classe (requis) android:name=".MyBR"
  - android:enabled=true pour rendre ce BR actif
  - android:exported=true pour autoriser ce BR à recevoir des intents en provenance de l'extérieur (autres applications, système)
- lorsque les BR sont déclarés statiquement, les BR sont enregistrés au moment du boot ou lorsque l'app est installée.

Broadcast Receiver

Liste des BroadcastReceiver enregistrés

- dans un terminal:

  adb shell dumpsys activity b

- pour une vue plus complète vous pouvez utiliser l'option `-h` option au lieu de `b`.

Broadcast Receiver

Comment les BR sont-ils avertis ?

$\Rightarrow$ BR s'enregistre auprès du système pour une liste d'intent-filters et...

1. à chaque diffusion d'un intent le système examine l'action associée à l'intent et le délivre à tous les BR qui se sont enregistrés pour cette action particulière
2. la délivrance de cet intent se fait par un appel à la méthode onReceive(Context,Intent) implémentée par chaque BR

Exemple : écouter les changements d'état du GPS

- notre application souhaite être avertie à chaque changement de l'état du GPS (allumé/éteint)
- nous créons une Application qui intègre un BroadcastReceiver pour l'action concernant le changement d'état du GPS

l'action pour l'intent-filter est :

action android:name="android.location.PROVIDERS_CHANGED"

Exemple : écouter les changements d'état du GPS

```xml
<receiver android:name=".ExampleOfBR">
    <intent-filter>
        <action android:name="android.location.PROVIDERS_CHANGED" />
    </intent-filter>
</receiver>
```

```java
public class ExampleOfBR extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        Toast.makeText(context, text: "GPS has changed!!",Toast.LENGTH_SHORT).show();
    }

}
```

Broadcast Receiver

Enregistrement dynamique

- création d'un IntentFilter
- création dynamique d'un BroadcastReceiver et enregistrement dynamique par un appel à la méthode registerReceiver() pour laquelle nous avons deux implémentaitons différentes :
  - la première proposée par LocalBroadcastManager pour les messages à l'échelle de l'application
  - la seconde proposée par Context pour les messages *system-wide*
- bonnes pratiques : l'enregistrement/désenregistrement doit être fait dans les méthodes onResume() et onPause() du cycle de vie

Broadcast Receiver

IntentFilter

- création: new IntentFilter()
- ajout de l'ensemble des actions d'intérêt : IntentFilter.addAction(...)
- autant de fois qu'il y a d'actions ciblées

registerReceiver

- méthode de la classe Context (dans le cas général)
- deux paramètres :
  - le BroadcastReceiver
  - l'IntentFilter

si l'IntentFilter a plusieurs actions, elles peuvent être distinguées par la méthode getAction() invoquée sur l'intent passé en paramètre de la méthode de callback BroadcastReceiver.onReceive()

Exemple : écouter les changements d'état du WIFI

- notre application a besoin de connaître les changements d'états du wifi
- un BroadcastReceiver est créé au sein de l'Activity
- lorsqu'un changement de l'état du wifi est diffusé, l'application réagit en conséquence

```java
public class MainActivity extends AppCompatActivity {

    TextView tv;
    BroadcastReceiver wifiBR;
    IntentFilter wifiIF;
    int nbOfChanged = 0;

    protected void onCreate(Bundle sis) {
        super.onCreate(sis);
        setContentView(R.layout.activity_main);
        tv = (TextView)findViewById(R.id.tv);
        wifiBR = new BroadcastReceiver() {
            @Override
            public void onReceive(Context context, Intent intent) {
                tv.setText("Wifi has changed !! "+nbOfChanged);
                nbOfChanged++;
            }
        };
        wifiIF = new IntentFilter();
        wifiIF.addAction("android.net.wifi.WIFI_STATE_CHANGED");
        wifiIF.addAction("android.net.wifi.STATE_CHANGE");
    }

    protected void onResume() {
        super.onResume();
        registerReceiver(wifiBR,wifiIF);
    }

    protected void onPause() {
        super.onPause();
        unregisterReceiver(wifiBR);
    }

}
```

Diffuser un événement

Comment les événements sont-ils diffusés aux BR?

- si votre application a besoin de diffuser un message, cela peut être fait par un appel à la méthode sendBroadcast(intent)
- cette méthode diffuse l'intent à tous les BR qui se sont enregistrés pour l'action correspondante

Diffuser un événement

- pour diffuser un événement l'application doit construire un intent et doit indiquer a minima l'action associée à l'intent de manière à ce que les BR puissent s'enregistrer pour cette action
- le champ action de l'intent est utilisé pour identifier de manière unique l'événement associé à cette diffusion
- la manière de procéder est très proche de celle qui est utilisée pour démarrer une nouvelle activité mais au lieu d'invoquer la méthode `startActivity(Intent)`, on invoque la méthode sendBroadcast(Intent)

Exemple : diffuser un événement

- l'exemple est composé :
  - d'une Activity intégrant un TextView et un Button,
  - d'un BroadcastReceiver
- Principe :
  - une fois pressé, le bouton déclenche la diffusion d'un intent dont l'action est MY_EVENT
  - lorsque le BR est notifié de cet événement il affiche un Toast indiquant qu'il a reçu cet événement

Activity

```
public class MainActivity extends AppCompatActivity {

    public final static String MY_EVENT = "mobapp.example.MY_EVENT";
    TextView tv;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        tv = (TextView)findViewById(R.id.tv);
    }

    public void broadcast(View v) {
        Intent broadcastIntent = new Intent(MY_EVENT);
        sendBroadcast(broadcastIntent);
        tv.setText("Broadcast done");
    }
}
```

BroadcastReceiver

```
public class BReceiverOfMyEvent extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        String actionReceive = intent.getAction();
        Toast.makeText(context, text: "Action received:"+actionReceive,Toast.LENGTH_SHORT).show();
    }
}
```

AndroidManifest

```
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="S4nov2017LLBRSender"
    android:roundIcon="@mipmap/ic_launcher_round"
    android:supportsRtl="true"
    android:theme="@style/AppTheme">
    <activity android:name=".MainActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <receiver android:name=".BReceiverOfMyEvent">
        <intent-filter>
            <action android:name="mobapp.example.MY_EVENT" />
        </intent-filter>
    </receiver>
</application>
```

Exercice : ping pong

- créez deux projets Android : Ping et Pong
  - le projet Ping est composé de deux classes java : une Activity et un BroadcastReceiver
  - le BroadcastReceiver est déclaré dans le fichier AndroidManifest et l'<action> de son IntentFilter est mobapp.example.PONG_EVENT
  - le projet Pong est composé d'une seule classe java : une Activity qui déclare dynamiquement un BroadcastReceiver enregistré avec un IntentFilter dont l'action est mobapp.example.PING_EVENT

Question

En vous basant sur ce que nous avons vu programmez les deux applications de telle sorte que lorsqu'on lance l'application Ping, des Toast alternant les messages Ping et Pong sont affichés à l'écran.

Remarques

- lorsque vous écrivez le code de la méthode onReceive(), gardez à l'esprit que les BR n'ont pas de thread qui leur soit dédié
- ainsi Android peut devoir démarrer un thread pour exécuter le code associé au BR
- par ailleurs, lorsque la méthode onReceive() est exécutée, le processus associé est de haute priorité

  $\Rightarrow$ vous ne devez pas implémenter de longues opérations dans cette méthode
- si vous avez besoin d'exécuter une longue opération, vous devez considérer une autre architecture pour votre application de telle sorte que votre longue opération soit exécuter dans le contexte d'un thread indépendant.