

# TPB3 : DIALOGUE ENTRE ACTIVITY

## Objectif du TP :

- Modification de l'activity MainActivity pour créer un cours Collectif ou Individuel
- Appel entre activity
- Echange d'informations d'une activity à une autre.

## MODIFICATION DU MAINACTIVITY

### 1) Modification de la vue de l'activité (activity\_main.xml)

Sur le même principe que le TP précédent, faites en sorte que l'activité principale puisse créer un nouveau cours comme ci-contre :

#### rappel :

- le cours (ABSTRACT) et non instanciable, il faut soit instancier un cours collectif, soit un cours individuel  

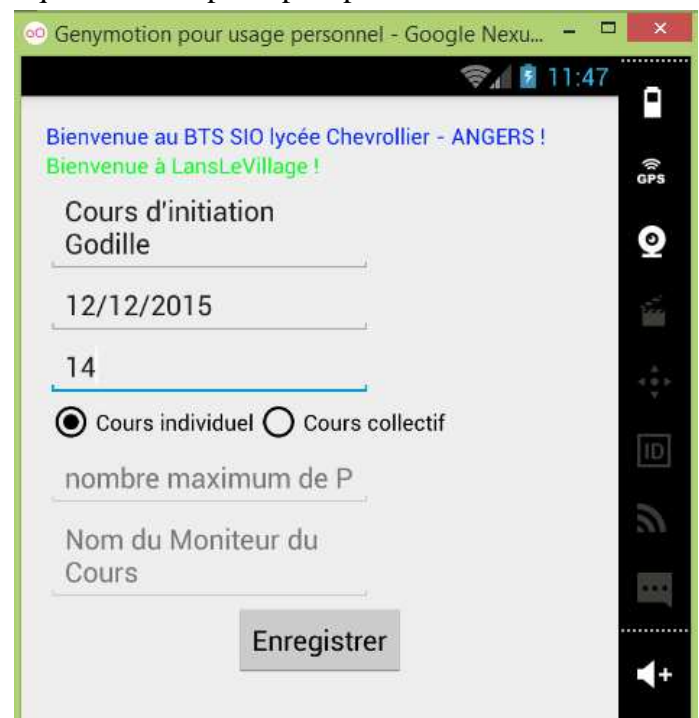
```
public abstract class Cours {  
    protected String intituleCours;  
    protected Date dateCours;  
    protected Integer heureCours;  
}
```
- Pour le cours Collectif  

```
public class CoursCollectifs extends Cours {  
    private byte nombreMax;  
    private ArrayList<Participant> lesParticipantsInscrits ;  
}
```
- Pour le cours individuel  

```
public class CoursIndividuels extends Cours {  
    private Participant leMoniteur;  
    private Participant unSeulParticipant;  
}
```

#### Conseils :

- Utiliser le mode Text pour paramétrer ces contrôles en copiant/collant ceux de l'activityParticipant. *Pour une fois que je vous autorise le copier/coller ;-)*
- Le nombre Max de participant et le nom du Moniteur seront **désactivés** au départ.
- Pour simplifier les cours individuels, on rentrera juste le nom du Moniteur (donc, il faut faire une surcharge de constructeur dans la classe participant afin de pouvoir créer un participant juste avec son nom).
- Pour les participants que ce soit pour un cours collectif ou individuel, on ajoutera les participants à l'aide de l'autre activité (ActivityParticipant) → plus tard !



Lancer votre application sur votre téléphone ou AVD en faisant Run sur la classe MainActivity

## 2) Gestion de la vue de l'activité (MainActivity.java)

Le but est :

- de gérer le clic des boutons radios afin d'afficher soit le nombre max de participant, soit le nom du moniteur.
- De gérer le clic du bouton enregistrer
- De créer suivant le cas, soit un cours collectif soit un cours individuel

### 1) Gérer le clic des boutons radio

**Rappel :** Tout objet GRAPHIQUE possède un id que l'on va pouvoir utiliser pour l'identifier à coup sûr ! (même principe qu'en Javascript ;)



**TRAVAIL à faire :** dans le fichier MainActivity.java

Dans la méthode onCreate() : lors de la création de l'activité

- Récupérer le bouton radio (RadioButton) par son id dans la vue courante  
`RadioButton rbC = (RadioButton) findViewById(R.id.radioButtonCoursC);`
- On paramètre le bouton radio pour lui dire qu'il va être « écouter »!  
`rbC.setOnClickListener(click);`

EN DESSOUS de la méthode onCreate() :

- On va « écouter » si ce bouton est cliqué ! Création de l'objet btnclick qui va réellement être dans l'écoute du click.

//A COMPLETER pour gérer tous les click sur cette Vue (View)

```
private OnClickListener click = new OnClickListener() {  
    //redéfinition de la méthode onClick  
    public void onClick(View v) {  
        switch (v.getId()) {  
            case R.id.radioButtonCoursC :  
                //A COMPLETER pour la gestion du click du radiobouton Cours collectif  
                break;  
            case R.id.radioButtonCoursI :  
                //A COMPLETER pour la gestion du click du radiobouton Cours individuel  
break;  
            case R.id.buttonEnregistrerCours :  
                //A COMPLETER pour la gestion du click du bouton  
                Cours c; //variable cours qui va soit être un cours collectif , soit  
                être un cours individuel → POLYMORPHISME  
                Toast.makeText(getApplicationContext(), "Enregistrement du Cours",  
                    Toast.LENGTH_SHORT).show();  
                break;  
        }  
    }  
};
```

## APPEL D'UNE AUTRE ACTIVITÉ EN LUI PASSANT DES DONNEES.

### Souhait :

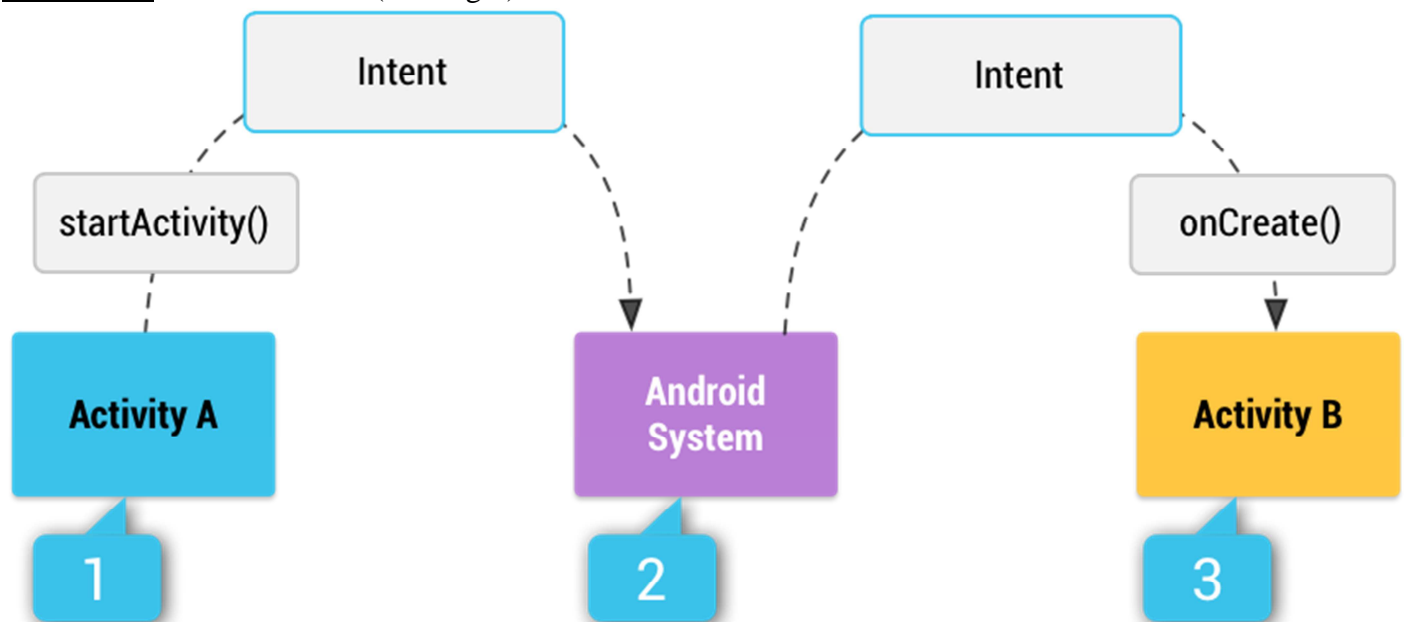
On souhaite que la MainActivity appelle l'activité participant en lui passant le cours qu'elle vient de créer. Cet appel se fera juste après la création du cours.

Comme cela le participant créé dans l'activité participant pourra être rajouté au cours.

### Contrainte :

Comme chaque activité est indépendante, le nouveau cours n'est pas connu de l'activité Participant. Donc il faut lui trouver un moyen de lui passer cette information.

### Solution 1 : Notion d'Intent (Messenger) :



An Intent is a messaging object you can use to request an action from another app component. Although intents facilitate communication between components in several ways like **to start an Activity** :

An Activity represents a single screen in an app. You can start a new instance of an Activity by passing an Intent to `startActivity()`. The Intent describes the activity to start and carries any necessary **data**.

*Quand on démarre une activité, on peut utiliser un « Messenger » qui peut contenir des informations. On pourra donc lui passer des informations à l'autre activité (celle que l'on appelle par le `startActivity()`)*

### Principe pour passer le Intent :

```
public class MainActivity extends Activity {

    /*déclaration de la variable Messenger*/
    private Intent i;

    /*Lors du click du bouton Enregistrer mais après avoir créé notre cours*/
    case R.id.buttonEnregistrer:
        /*démarrage d'une activité simplement en lui passant des informations dans le
        intent*/
        Toast.makeText(getApplicationContext(), "clic sur Activité
        2", Toast.LENGTH_SHORT).show();
        i = new Intent(getApplicationContext(), Activity2.class);
        i.putExtra("info1", "Démarrage de l'activité");
        i.putExtra("info2", 2);
        i.putExtra("info3", monCours.toString());
        startActivity(i);

        break;
```

### Principe pour récupérer le Intent dans l'autre Activity:

```
public class Activity2 extends Activity {

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_activity2);

        /*création du bundle qui va réceptionner les données passées par le Intent*/
        Bundle b = getIntent().getExtras();

        String info1 = b.getString("info1");
        int info2 = b.getInt("info2");
        String info3 = b.getString("info3");

        Toast.makeText(getApplicationContext(), info1 + " : " + Integer.toString(info2),
        Toast.LENGTH_LONG).show();
```



### TRAVAIL à faire : dans le fichier MainActivity.java

- Créer un nouveau bouton nommé « enregistrement des participants »
- Créer l'intent
- Passer lui le cours
- Appeler la nouvelle activité
- Récupérer ces infos dans l'autre activité

### Problème :

- Le Messenger(Intent) ne sait pas passer des **Objets** (juste des chaînes, entier etc..).
- A part afficher le cours (chaîne), la nouvelle activité (ActivityParticipant) ne peut pas réellement le récupérer !
- Comment faire ? : La **sérialisation** ....

## LA SÉRIALISATION (SÉRIALISER MON COURS POUR LE TRANSMETTRE À L'AUTRE ACTIVITY)

### QU'EST-CE QUE LA SÉRIALISATION DES OBJETS ?

La sérialisation d'un objet est le processus de sauvegarde d'un objet complet sur fichier, d'où il pourra être restauré à tout moment. Le processus inverse de la sauvegarde ( restauration ) est connu sous le nom de désérialisation.

<https://openclassrooms.com/courses/la-programmation-reseau-en-net/la-serialisation-des-objets>

#### Avantages majeurs :

- Très utilisé pour transmettre des données sur un réseau ou entre applications.
- simple fichier texte donc le transfert est facile

#### Inconvénient majeur :

- Si la classe change (ajout d'attributs par exemple) alors nous ne pouvons plus récupérer les objets sérialisés.

#### Comment mettre en place la sérialisation ?:

Il faut marquer les classes que l'on veut sérialiser en écrivant sur chaque classe `implements Serializable`

```
public class Cours implements Serializable {  
public class CoursCollectif extends Cours implements Serializable {  
public class CoursIndividuel extends Cours implements Serializable {  
public class Participant implements Serializable {
```

puis il faut importer la bibliothèque : `import java.io.Serializable;`

*Que dois-je sauvegarder ?*

**Je dois sauvegarder un cours qu'il soit collectif ou individuel et les futurs participants !**

```
//envoyer le messenger avec l'objet Serialisé  
i = new Intent(getApplicationContext(),ActivityParticipant.class);  
i.putExtra("coursCree", (Serializable)c);  
startActivity(i);  
break;
```

```
//recupéré le messenger  
Cours coursRecupere = (Cours) getIntent().getSerializableExtra("coursCree");
```

**i** La sérialisation est souvent utilisée pour **transmettre des objets sur une connexion réseau** afin de les passer à une autre application par exemple. Le fait de les sérialiser, ces objets encombrant moins la bande passante.

**i** L'idéal est d'utiliser l'interface **Parcelable** au lieu de **Serializable** mais c'est un peu plus compliqué!

#### **Avis à ceux qui sont intéressés !**

`Serializable` is slow on Android. Borderline useless in many cases in fact.

`Parcel` and `Parcelable` are fantastically quick, but its [documentation](#) says you must not use it for general-purpose serialization to storage, since the implementation varies with different versions of Android (i.e. an OS update could break an app which relied on it).

<http://stackoverflow.com/questions/5550670/benefit-of-using-parcelable-instead-of-serializing-object>

---

**Profiter de l'objet Cours Récupéré pour ajouter les participants !**



**TRAVAIL à faire :** dans le fichier `ActivityParticipant.java`

- Sur le clic du bouton Enregistrer :
  - Ajouter le nouveau participant au cours que l'on vient de récupérer
  - Remettre les zones de texte (`editText`) à blanc !
- Sur un bouton liste des Participants :
  - Afficher Tous les participants de ce cours.
- Sur un bouton Sortir :
  - Quitter l'activity : `finish()` ;
- TESTS :
  - Faire les tests nécessaires pour l'ajout de plusieurs participants pour le cours individuel.
    - Que se passe-t-il ?
  - Faire les tests nécessaires pour l'ajout de plusieurs participants pour le cours collectif (dépassement du nombre max de participant).
    - Que se passe-t-il ?



**TRAVAIL à faire EN PLUS :**

- créer une nouvelle activité *ActivitySaison* pour la création d'une nouvelle saison (en instanciant la classe *PlanningSaison*).
- faire en sorte que cette activité se lance au démarrage de votre application.
- une fois la saison créée, appeler l'activité *MainActivity* (qui gère les différents cours). ces cours viendront s'ajouter à la liste des cours dans la classe *PlanningSaison*.