ISET SFAX

AU 2020/2021 SEMESTRE I

Devoir Surveillé		
Classe : *SEM31*	***Matière : D****éveloppement* ***M****obile* ***A****vancé*	***Nb pages :*** *6*
Documents Non Autorisés	***Enseignant :*** *Souissi Hafedh*	***Durée :*** *1 heure*
Barème : 20 = 10 + 10		

N.B. : - Ne donner que le code des méthodes demandées.

- Les deux exercices sont indépendants

Exercice1 (10 points)

« **SuiviVoitures** » est une application Android qui permet à une agence de location de voitures de faire le suivi des entretiens de ses voitures. « **SuiviVoitures** » permet d'ajouter un ensemble de voitures à une base de données SQLite "suivi.db" et d'enregistrer la date du dernier entretien ainsi que son prix. Une voiture est définie par un id (id), un matricule (matricule), une marque (marque) et une couleur (couleur), une date du dernier entretien (dateDE) et un prix du dernier entretien (prixDE).

« **SuiviVoitures** » contient trois activités « MainActivity », « Nouveau » et « Modification ». Le code de ces classes et leurs interfaces sont donnés dans « Annexe1 ».

1- Donner le code des méthodes **ajouter()** de la classe « **Nouveau** » qui permet d'ajouter la voiture saisie et d'effacer les EditTexts (la valeur par défaut de dateDE= "" et celle de prixDE est 0).

2- Donner le code de la méthode **remplir ()** de la classe « **Modification** » qui permet de remplir spVoiture par les voitures de la table "Voiture" en utilisant la requête « Select * From Voiture ».

3- Donner le code de la méthode **actualiser()** de la classe « **Modification** » qui permet d'afficher la dateDE (date Dernier Entretien) et le prixDE (prix Dernier Entretien) de la voiture sélectionné dans spVoiture dans les EditTexts.

4- Donner le code de la méthode **modifier()** de la classe « **Modification** » qui permet de modifier dans la table "Voiture" de données la dateDE (date Dernier Entretien) et la prixDE (prix Dernier Entretien) de la voiture sélectionné dans spVoiture.

Exercice2 (10points)

« **ReservationVoiture** » est une application Android qui permet à son utilisateur de réserver une voiture chez une agence de location de voitures. Une voiture est définie par un id (id), un matricule (matricule), une marque (marque) et une couleur (couleur).

« **ReservationVoiture** » contient une activité « MainActivity » et une classe « Voiture ». Le code de ces classes et l'interface sont donnés dans « Annexe2 ».

« **ReservationVoiture** » appelle une application Web hébergées dans l'adresse « http://192.168.10.19 :80/ReservationVoiture/ » et elle contient deux pages « ListeVoitures.php », et « Reservation.php » qui utilisent la méthode « POST ».

« ListeVoitures.php » ne prend aucun paramètre et retourne un objet JSON qui contient toutes les voitures sous la forme suivante :

```
{"voitures":[
  {"id":"1","matricule":"1234 Tunisie 240",
   "marque":"BMW", "couleur":"Bleu"},
  {"id":"2","matricule":"1235 Tunisie 240",
   "marque":"Mercedes", "couleur":"Gris"},

]}
```

« Reservation.php » prend **quatre paramètres** qui sont le id de la voiture (id), le CIN du client (cin), la date de début de la réservation (dateD) et le nombre de jours de la réservation(nbJ), elle ajoute la réservation à la table « Reservation » de la base MySQL et retourne
{" ETAT" : "SUCCES"} en cas de succès ou {**" ETAT" : "ECHEC"}** en cas d'échec.

1- Donner le code de la méthode **Remplir()** de la classe « **MainActivity** » qui permet de :

   a. d'appeler la page ListeVoitures.php,

   b. d'analyser la réponse JSON

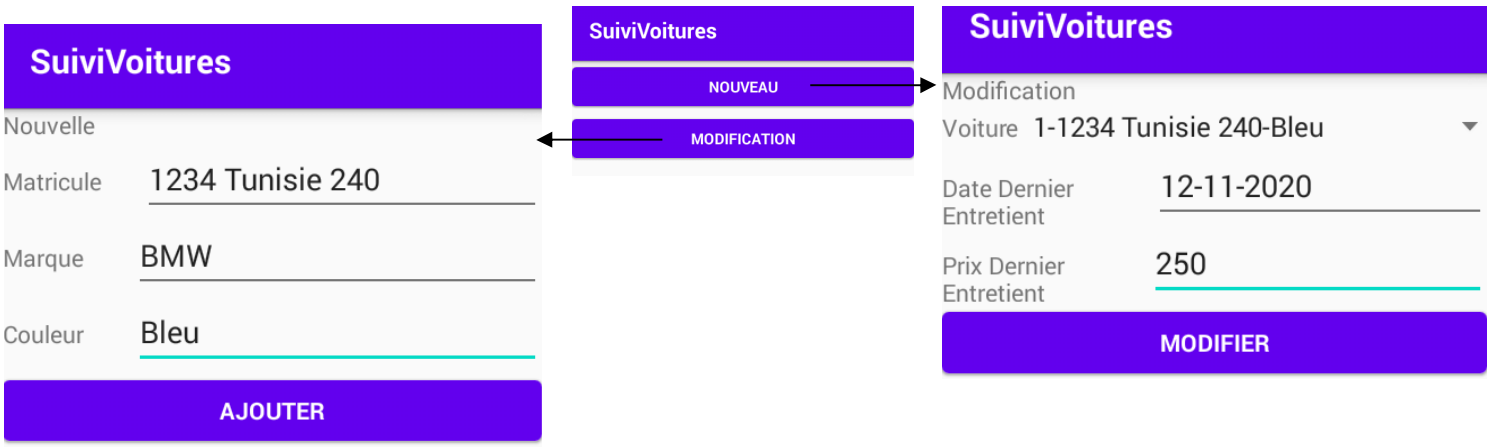
   c. de remplir le Spinner spVoiture.

2- Donner le code de la méthode **reserver()** de la classe « **MainActivity** » qui permet de :

   a. appeler la page Reservation.php en passant les paramètres nécessaires.

   b. Si la réservation réussi alors elle efface les trois EditText et place le curseur dans edCIN sinon elle affiche un message d'erreur (Toast).

Annexe1



SQLite

Insertion et modification

```
SQLiteBib b = new SQLiteBib(this, "base.db", null, 1);
SQLiteDatabase db;
db = b.getWritableDatabase();
ContentValues v = new ContentValues();
v.put("nb", Integer.parseInt("10"));
db.insert("livre", null, v);
db.update("livre",v,"id=?",new String[]{"1"});
db.close();
```

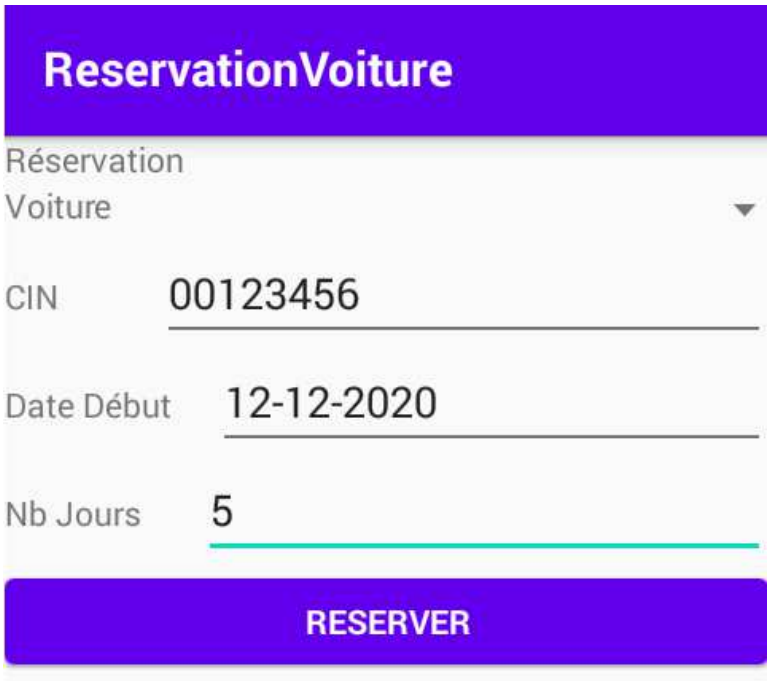
Selection

```
SQLiteBib b = new SQLiteBib(this, "bib.db", null, 1);
    SQLiteDatabase db = b.getWritableDatabase();
      Cursor c = db.rawQuery("Select * from livre where nbPage>50;",null});
    adpLivre.clear();
    while (c.moveToNext()) {
      int id = c.getInt(0); ...
      adpLivre.add(...);
    }
```

```
package com.suivi;
public class Voiture {
    private int id;
    private String matricule;
    private String marque;
    private String couleur;
    private String dateDE;
    private int prixDE;
    public Voiture(int id, String matricule, String marque,
                   String couleur, String dateDE, int prixDE) {
        this.id = id;                this.matricule = matricule;
        this.marque = marque;        this.couleur = couleur;
        this.dateDE = dateDE;        this.prixDE = prixDE;
    }
    public int getId()           {       return id;        }
    public String getMatricule() {       return matricule; }
    public String getMarque()    {       return marque;    }
    public String getCouleur()   {       return couleur;   }
    public String getDateDE()    {       return dateDE;    }
    public int getPrixDE()       {       return prixDE;    }
    @NonNull
    @Override
    public String toString() {
        return id+"-"+matricule+"-"+couleur;
    }
}
```

```
package com.suivi;
public class SQLiteSuivi extends SQLiteOpenHelper {
    public SQLiteSuivi(Context context, String name, SQLiteDatabase.CursorFactory factory,
                       int version) {
        super(context, name, factory, version);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        String sql="create table Voiture (id INTEGER PRIMARY KEY AUTOINCREMENT,matricule
text NOT NULL,marque text NOT NULL,couleur text NOT NULL,dateDE text NOT NULL,prixDE
INTEGER );";
        db.execSQL(sql);
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }
}
package com.suivi;
public class MainActivity extends AppCompatActivity{

    ...
}
package com.suivi;
public class Nouveau extends AppCompatActivity {
    private EditText edMatricule;
    private EditText edMarque;
    private EditText edCouleur;
    private Button btnAjouter;

    ...

    private void ajouter() {
        // Permet d'ajouter la voiture saisie à la base SQLite,
        // efface les trois EdiText


    }

}
package com.suivi;
public class Modification extends AppCompatActivity {
    private Spinner spVoiture;
    private EditText edDateDE;
    private EditText edPrixDE;
    private Button btnModifier;
    private ArrayAdapter<Voiture> adpVoiture;
    ...

    private void remplir() {
        //Permet de remplir le Spinner spVoiture par les voitures de la table.

    }

    protected void actualiser() {
        //Permet d'afficher dans edDateDE et edPrixDE les valeurs relatives à
        //la voiture sélectionnée dans le spinner spVoiture.
    }

    private void modifier() {
        //Si le Spinner spVoiture contient une sélection et si edPrixDE ne contient pas
        //une chaîne vide, alors elle modifie la date du dernier entretient et
        //son prix pour la voiture sélectionnée dans le spinner spVoiture

    }

}
```

Annexe2



Volley

```
RequestQueue queue = Volley.newRequestQueue(this);
String url = "http://100.20.12.2:80/Page.php";
StringRequest sr = new StringRequest(Request.Method.POST,
     url, new Response.Listener<String>() {
       @Override
       public void onResponse(String response) {
              //Traitement de la réponse
       }
     }, new Response.ErrorListener() {
          @Override
          public void onErrorResponse(VolleyError e) {
              //Traitement de l'erreur
          }
        }) {
  @Override
  public Map<String, String> getParams() throws AuthFailureError {
    HashMap<String, String> headers = new HashMap<String, String>();
    headers.put("param", "val");
    return headers;
  }
  };
  queue.add(sr);
```

JSONObject

```
JSONObject json = new JSONObject("{\"c1\":\"v1\",\"c2\":\"12\"\",\"c3\":\"12.4\"}");
String s = json.getString("c1");
int s = json.getInt("c2");
float s = Float.parseFloat(json.getString ("c3"));
```

JSONArray

```
try {
      JSONArray a = json.getJSONArray("liste");
      for (inti = 0; i<aLivre.length(); i++) {
            JSONObject o = a.getJSONObject(i);
      }
} catch (JSONException e) {
      e.printStackTrace();
}
```

```
package com.reservation;
public class Voiture {
    private int id;
    private String matricule;
    private String marque;
    private String couleur;
    public Voiture(int id, String matricule, String marque, String couleur) {
        this.id = id;                this.matricule = matricule;
        this.marque = marque;        this.couleur = couleur;
    }
    public int getId()           {        return id;        }
    public String getMatricule() {        return matricule; }
    public String getMarque()    {        return marque;    }
    public String getCouleur()   {        return couleur;   }
    @NonNull
    @Override
    public String toString() {
        return id+"-"+matricule+"-"+couleur;
    }
}
```

```
package com.reservation;
//imports
public class MainActivity extends AppCompatActivity {
  private Spinner spVoiture;
  private EditText edCIN;
  private EditText edDateD;
  private EditText edNbJ;
  private Button btnReserver;
  private ArrayAdapter<Voiture> adpVoiture;
  ...
```

```
protected void remplir() {
    //Permet d'appeler la page ListeVoitures.php,
    //d'analyser la réponse JSON
    //et de remplir le Spinner spVoiture.



}
```

```
protected void reserver() {
    //Permet de faire une reservation de la voiture pour le client
    //en appelant la page Reservation.php et en passant les paramètres nécessaires.
    //Si la réservation réussi alors elle efface les trois EditText
    //sinon elle affiche un message d'erreur (Toast).

}
```

```
}
```