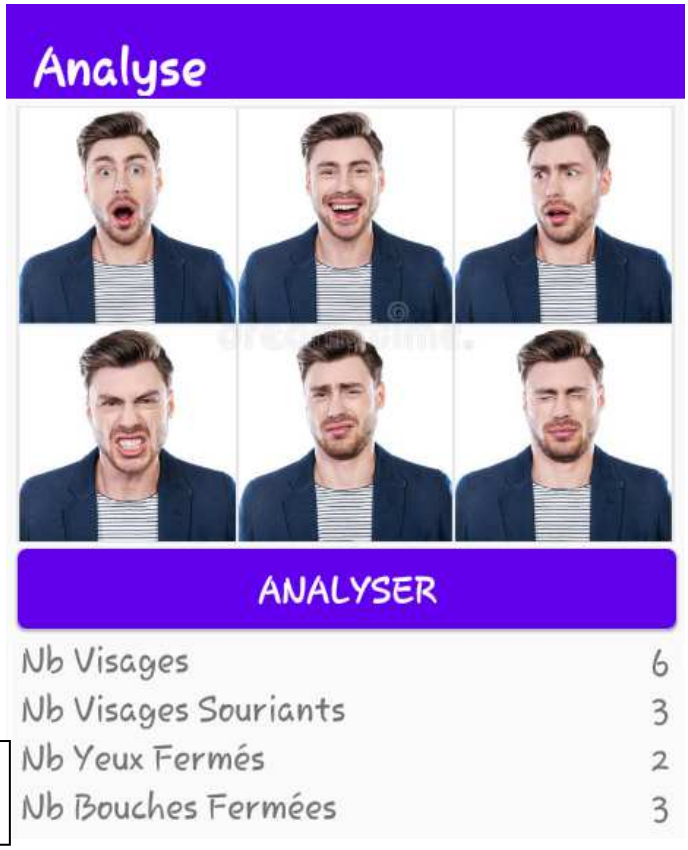
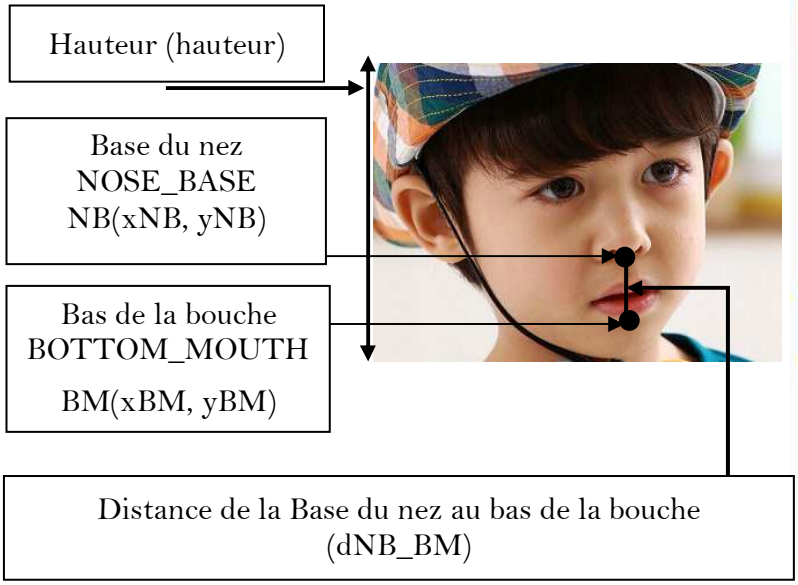
ISET SFAX AU 2022/2023 SEMESTRE I		
DEVOIR FINAL		
Classe : SEM31	*Matière : **D**éveloppement **M**obile **A**vancé*	*Nb pages : 13*
*Enseignant : **S**ouissi **H**afedh*		
Documents Non Autorisés	*Barème : 20 = 5 + 8 + 7*	*Durée : 1 heure 30 minutes*

Exercice1 (Utiliser Annexe I)

"Analyse" est une application Android qui permet de détecter les visages d'une image et d'afficher un ensemble de statistiques sur les visages détectés. **"Analyse"** contient une seule activité.





Pour obtenir la distance "dNB_BM " entre les deux points NB(xNB, yNB) et BM(xBM, yBM), utiliser la formule : **dNB_BM=Math.sqrt(Math.pow(xNB - xBM, 2)+Math.pow(yNB - yBM, 2))**.

Le coefficient **cNBMBH = dNB_BM/hauteur** permet de savoir si la bouche est fermée ou non : Si **cNBMBH < 0,27** alors la bouche est fermée.

La méthode "analyser()" permet de détecter les visages de l'image, de calculer et d'afficher :

- le nombre de visages,
- le nombre de visage souriant (la probabilité qu'il rit est strictement supérieure à **0.6**),
- le nombre d'yeux droites et gauches fermés (la probabilité que l'œil est ouverte est strictement inférieur à **0.5**),
- le nombre de bouches fermées.

1- Donner le code de la méthode **"analyser()"** de la classe **"MainActivity"**.

Exercice2 (Utiliser Annexe II)

L'objectif de cet exercice est de réaliser une application android qui s'utilise dans un bateau pour connaître l'état de la mer en fonction des accélérations appliquées sur le bateau par les vagues et la pesanteur. L'état de la mer peut être "Mer Calme", "Mer Agitée" ou "Mer Très Agitée".

Le smartphone qui contient l'application est placé dans le bateau. C'est la norme des accélérations sur les trois axes (x, y et z) qui permet de déduire l'état de la mer. La norme de l'accélération est calculée par la formule : **normeAcc**=Math.sqrt(Math.pow(x,2)+Math.pow(y,2)+Math.pow(z,2))

Le tableau suivant présente l'état de la mer et l'image à afficher en fonction de la norme des accélérations :

Norme des accélérations (normeAcc)	**Etat**	**Image à afficher**	**Id de l'image à afficher**
normeAcc <13	Mer Calme		R.drawable.mer_calme
13<= normeAcc <17	Mer Agitée		R.drawable.mer_agitee
17<= normeAcc	Mer Très Agitée		R.drawable.mer_tres_agitee

"EtatMer" est une application Android qui permet d'afficher la norme des accélérations interceptées par le capteur d'accélération et de modifier l'image affichée suivant la valeur intercepté en respectant le tableau précédent.

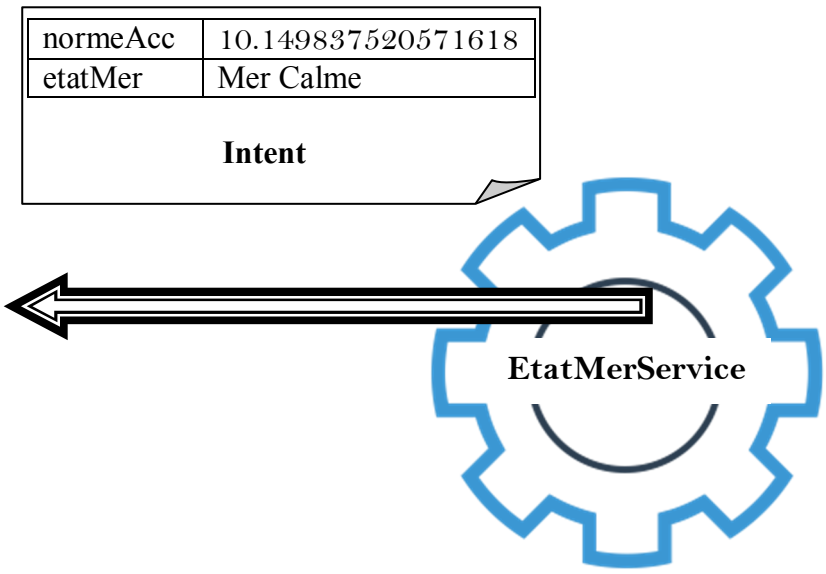
"EtatMer" est composée d'une activité **"MainActivity"** et d'un service **"EtatMerService"**. "EtatMerService" utilise un capteur de type TYPE_ACCELEROMETER pour intercepter les

modifications des accélérations sur les trois axes, par la suite il calcule la norme des accélérations et déduit l'état de la mer et diffuse (broadcast) ses deux valeurs dans un Intent.

Lorsque "MainActivity" fait un "Resume" elle démarre le service et lorsqu'elle fait une "Pause" elle arrête le service. "MainActivity" utilise un BroadcastReceiver pour lire la valeur "normeAcc" et "etatMer" envoyée par le service et de les afficher dans "tvEtat". Suivant la valeur de "normeAcc", "imgEtat" affiche IMG_MER_CALME ou IMG_MER_AGITEE ou IMG_MER_TRES_AGITEE.

Le texte affiché dans "tvEtat" est de la forme suivante : normeAcc => Etat, par exemple pour une normeAcc = 10.149837520571618 le texte affiché est "10.149837520571618 => Mer Calme".

Ci-dessous, un exemple de fonctionnement de l'application :



1- Donner le code des méthodes "ajouterEcouteur()", "demarrerService() ", "arreterService()", "onResume()", "onPause()" et "actualiser(…)" de la classe "**MainActivity**".

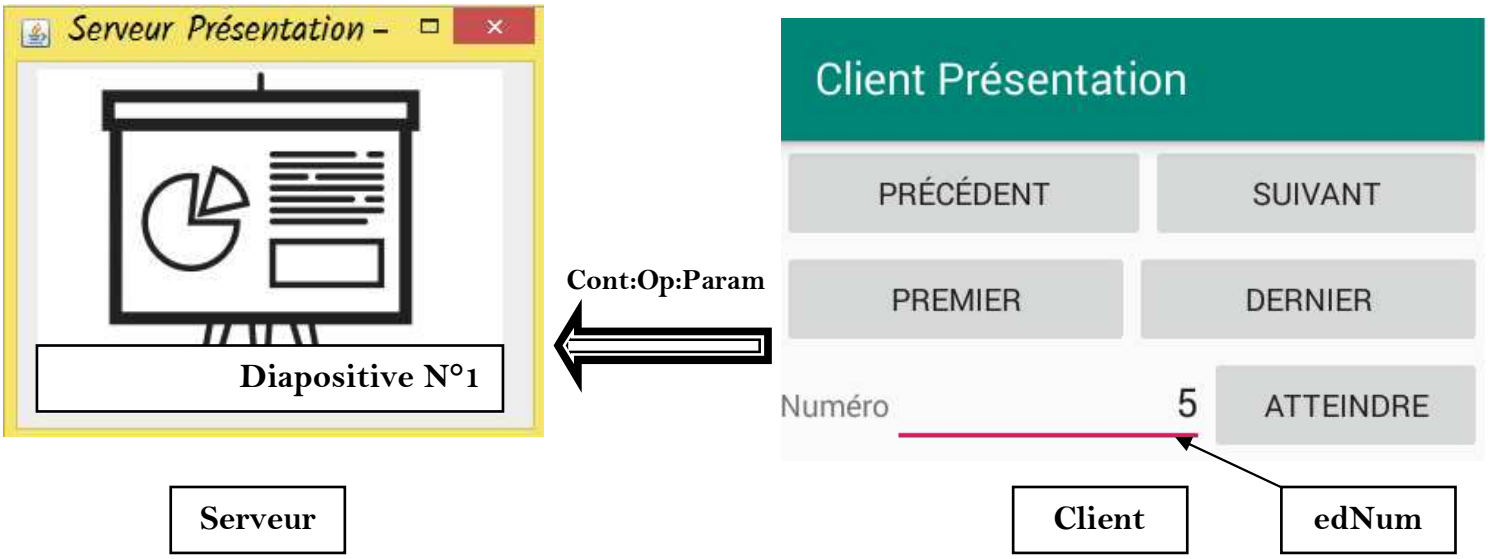
2- Donner le code des méthodes "init()", "onDestroy()" et de "onSensorChanged()" de la classe "**EtatMerService**".

Exercice3 (Utiliser Annexe III)

On désire développer une application qui permet de contrôler une présentation composée de plusieurs diapositives par un smartphone. Cette application utilise les sockets. Le programme serveur est installé sur un ordinateur d'adresse "192.46.10.20". Le programme client est installé sur un Smartphone android pour envoyer les commandes de l'utilisateur au serveur. Les contrôles qu'on peut effectuer sur la présentation et les messages envoyés par le socket clients sont résumés dans le tableau suivant :

Contrôle (Cont)	Opération (Op)	Paramètre (Param)	Message envoyé
PrecSuiv (0)	Précedent (0)		0:0
	Suivant (1)		0:1
PremDern(1)	Premier(0)		1:0
	Dernier (1)		1:1
Att (2)	Atteindre (0)	numero	2:0:numero

Le fonctionnement de l'application est comme suit :



1-Donner le code des méthodes " lancerThreadServeur()", "demarrerServeur()" et "executerCommande(…)" de la classe " **ServPresentation**" (coté serveur).

2-Donner le code des méthodes " lancerThreadClient()", "demarrerClient()", "envoyer() ", "precedent() ", "suivant() ", "premier() ", "dernier()" et "atteindre()" de la classe "**MainActivity**" (coté client).

Annexe I

Bitmap	//Pour transformer l'image d'un ImageView en bitmap Bitmap bitmap = ((BitmapDrawable)imgView.getDrawable()).**getBitmap**();
FaceDetector.Builder	FaceDetector.**Builder**(Context context); FaceDetector.Builder **setTrackingEnabled** (boolean trackingEnabled) FaceDetector.Builder **setLandmarkType** (int landmarkType) FaceDetector.Builder **setClassificationType** (int classificationType) FaceDetector **build**();
Frame	Frame frame = new Frame.Builder().setBitmap(bitmap).**build**();
FaceDetector	SparseArray<Barcode> **detect**(Frame frame); **ALL_LANDMARKS** **ALL_CLASSIFICATIONS**
SparseArray<Face>	int **size**() Face **valueAt**(int index);
Face	float **getHeight()** float **getIsSmilingProbability()** float **getIsRightEyeOpenProbability()** float **getIsLeftEyeOpenProbability()** List<Landmark> **getLandmarks**() (for (Landmark l : face.getLandmarks()) {})
Landmark	PointF **getPosition**() Retourne les coordonnées (x, y) du landmark avec (0, 0) est le point supérieur gauche de l'image. int **getType**() Retourne le type du landmark (NOSE_BASE, BOTTOM_MOUTH, etc.).
PointF	float **x** float **y**

```
package com.analyse;
//imports
public class MainActivity extends AppCompatActivity {
    private ImageView imgVisages;
    private Button    btnAnalyser;
    private TextView  tvNbV;       //TextView nombre de visages
    private TextView  tvNbVS;      //TextView nombre de visages souriants
    private TextView  tvNbYF;      //TextView d'yeux fermés
    private TextView  tvNbBF;      //TextView nombre de bouches fermées
    ...
    private void ajouterEcouteur() {
        //btnAnalyser.setOnClickListener(...) -> analyser();
    }

    private void analyser() {
    }

}
```

Annexe II

Activity
startService(Intent intent) registerReceiver(BroadcastReceiver br, new IntentFilter(String filter))
stopService(Intent intent) unregisterReceiver(BroadcastReceiver br)
Intent
Intent() Intent(Conext context, Class classe) setAction(String action)
putExtra(String nom, double valeur)
double getDoubleExtra(String nom, double defaultValue) String getDoubleExtra(String nom)
BroadcastReceiver
BroadcastReceiver br= new BroadcastReceiver() { @Override public void onReceive(Context context, Intent intent) { } };
Service
sendBroadcast(Intent intent)
SensorEventListener
@Override public void onAccuracyChanged(Sensor sensor, int accuracy) { } @Override public void onSensorChanged(SensorEvent event) { }
SensorManager
SensorManager smg = (SensorManager) getSystemService(SENSOR_SERVICE); SENSOR_DELAY_UI
Sensor
Sensor sensor = smg.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
SensorEvent
sensor.getType() event.values[int indice]
ImageView
imgArbre.setImageResource(IMG_ARBRE); // permet d'afficher l'image d'id IMG_ARBRE dans imgArbre

```
package com.mer;
//imports
public class MainActivity extends AppCompatActivity {
    public static final int IMG_MER_CALME       = R.drawable.mer_calme;
    public static final int IMG_MER_AGITEE      = R.drawable.mer_agitee;
    public static final int IMG_MER_TRES_AGITEE = R.drawable.mer_tres_agitee;
    private TextView          tvEtat;
    private ImageView         imgEtat;
    private BroadcastReceiver br;
    ...
    private void ajouterEcouteur()              {         }
    private void demarrerService()              {         }
    private void arreterService()               {         }
    @Override
    protected void onResume()                   {         }
```

```
    @Override
    protected void onPause()                    {         }
    protected void actualiser(Intent intent)    {         }
```

```
}
package com.mer;
//imports
public class EtatMerService  extends Service implements SensorEventListener {
    public static final String ACTION_ETAT_MER="ETAT_MER";
    private SensorManager smg;
    private Sensor        acc;
    @Override
    public IBinder onBind(Intent intent) {
        return null;
    }
    @Override
    public void onCreate() {
        super.onCreate();
        init();
    }
```

```
    private void init()                             {     }
    @Override
    public void onDestroy()                         {     }
    @Override
    public void onSensorChanged(SensorEvent event) {     }
```

```
    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) { }
}
```

Annexe III

ServerSocket
ServerSocket(int port) Socket accept()
Socket
Socket(Inet4Address i, int port) InputStream getInputStream() OutputStream getOutputStream()
Inet4Address
Inet4Address i = (Inet4Address) Inet4Address.getByName(ADR_SERVEUR);
Runnable

Runnable r = new Runnable() {
@Override
public void run() { }
};

Thread
Thread(Runnable r) start()
BufferedReader
IntputStream inp= … ;

BufferedReader br = new BufferedReader(new InputStreamReader(inp)); String cmd = br.readLine();
PrintWriter
OutputStream out= … ; PrintWriter pw = new PrintWriter(new BufferedWriter(new OutputStreamWriter (out)), true); String cmd= "Message" ; pw.println(cmd);
String
String[] split(String sep)

```
package com.serv.pres;
public class ServPresentation extends JFrame {
  private static final long serialVersionUID = 1L;
  private static final int PORT = 6020;
  private ServerSocket   ss;
  private Socket         s;
  private BufferedReader br;
  ...
  protected void precedent   ()      {  // affiche la diapositive précédente    }
  protected void suivant     ()      {  // affiche la diapositive suivante      }
  protected void premier     ()      {  // affiche la première diapositive      }
  protected void dernier     ()      {  // affiche la dernière diapositive      }
  protected void atteindre (int num) {  // affiche la diapositive de numéro num }
  private void lancerThreadServeur() {
  }
  private void demarrerServeur() {
  }
  private void executerCommande(String commande) {
  }
}
package com.client.pres;
//imports
public class MainActivity extends Activity {
  private static final String ADR_SERVEUR = "192.46.10.20";
  private static final int PORT = 6020;
  private Button      btnPrec;
  private Button      btnSuiv;
  private Button      btnPremier;
  private Button      btnDernier;
  private Button      btnAtteindre;
  private EditText    edNum;
  private Socket      s;
  private PrintWriter pw;
   ...
  private void lancerThreadClient()          {     }
  private void demarrerClient()              {     }
  private void envoyer(final String cmd)     {     }
  private void precedent   ()                {     }
  private void suivant     ()                {     }
  private void premier     ()                {     }
  private void dernier     ()                {     }
  private void atteindre   ()                {     }
}
```