3.4 The Accelerometer-Based Gestures Library

In this section, we explain how our implementation works and communicates with developer or anyone who is using our library. In this section we will briefly explain how to use library in an efficient and effective way.

We followed core concepts of object oriented model of programming to make sure everyone who knows basics of programming can easily access and use our library. We exposed our implementation through interface class named as **GestureDetectionInterface** with some methods that must be implemented while using library.

An example of code to implement interface is shown below:

```
1 public class DemoClass extends FragmentActivity implements
2 GestureDetectionInterface {
3   @Override
4        protected void onCreate(Bundle savedInstanceState) {
5        super.onCreate(savedInstanceState);
6        setContentView(R.layout.SomeLayout);
7        }
8 }
```

Your class must show error to implement three methods listed below in bullets. This class is to send messages from our library to the application which is using it. Methods that must be implemented are listed below with their description.

- **public void registrationError(String)**

  This methods is invoked if there is any problem in the beginning phase of registering library with your application. In case of any error this method is invoked and sends a Java string class object with details. Users can follow the error and retry after fixing the issue.

- **public void GestureType(String)**

  Once you have successfully bounded library with your application and it start working this method is invoked every time a gesture is recognized. GestureType method sends a Java String class object with type of gesture recognized. Your application can use **Global.Gestures.** type of string to match the gesture type.

- **public void continuosValues(String, double [ ], double [ ])**

  continuosValues method has two parameters which carries two Java double type arrays. This method is invoked in only two cases. One when application has already detected Extreme Zoom gesture and second is Drag gesture. First parameter carries 3

dimensional values of phone position at which gesture was detected and second double type array contains 3 dimensional values for current position of phone. This way you can track position of phone and perform actions accordingly to your application.

The code below shows an example of the class code after implementing the above methods:

```
1 public class DemoClass extends FragmentActivity implements
2 GestureDetectionInterface {
3        @Override
4        protected void onCreate(Bundle savedInstanceState) {
5        super.onCreate(savedInstanceState);
6        setContentView(R.layout.SomeLayout);
7        }
8        @Override
9        public void registrationError(String s) {
10      }
11       @Override
12  public void GestureType(String s) {
13       }
14       @Override
15       public void continuosValues(String s, double[] doubles, double[] doubles1) {
16       }
17 }
```

Once you implement interface you have to pass your application context to our gesture detection class to start the application process. You can start receiving gesture recognition anytime by sending application context to the constructor of **DetectGestures** class available in the library. You can do this by just invoking the constructor from Main Activity class of your application. Code for invocation is show below.

```
1 Context _context =  getApplicationContext();
2 DetectGestures _objDG = new DetectGestures(this,_context);
```

- **public DetectGestures(GestureDetectionInterface, Context)**

  DetectGesture method receives two objects as parameter. First parameter must be the object of your main activity class which can be sent by just typing **this** word as parameter value. Second parameter is application context. You would need to make an object of your application context and send it as parameter to this method. If you do everything as explained you application would start working successfully and if not then you will receive an error message through our **registrationError(String)** method which is already been explained.

Here is an example of the clasee code after implmenting the method:

```
1 public class DemoClass extends FragmentActivity implements
2 GestureDetectionInterface {
3 @Override
4 protected void onCreate(Bundle savedInstanceState) {
5               super.onCreate(savedInstanceState);
6               setContentView(R.layout.SomeLayout);
7               Context _context =  getApplicationContext();
8               DetectGestures _objDG = new DetectGestures(this,_context);
9       }
10      @Override
11      public void registrationError(String s) {
12      }
13      @Override
14      public void GestureType(String s) {
15      }
16      @Override
17      public void continuosValues(String s, double[] doubles, double[] doubles1) {
18      }
19 }
```

Third class is global class which is used for maintaining the communication standards between library and application. We have defined standard communication model for better understanding and to avoid complications. You can access them anytime withing your application by accessing **Global** class without making any intent. Messages are divided in three different categories. First category ca be accessed by typing **Global.** and then any of name listed in the first category. Below are the standard messages of category 1 which can be used anytime depending on their data types .

- **int X = 0**
- **int Y = 1**
- **int Z = 2**
- **int X_AXIS = 0**
- **int Y_AXIS = 1**
- **int Z_AXIS = 2**
- **String WAIT_MESSAGE = "Wait"**

Second category is based on error messages which can be accessed by using **Global.Errors.** and name of variable listed below

- **String ERROR_REGISTRATION = "Sensor Manager Registration Error"**

Third category is for Gesture messages and very important one. It can be accessed by using **Global.Gestures.** and then name of any gesture listed below.

- **String TAP = "Tap"**
- **String PRESS = "Press"**
- **String ZOOM_IN = "ZoomIn"**
- **String ZOOM_OUT = "ZoomOut"**
- **String LEFT_DRAG = "LeftDrag"**
- **String RIGHT_DRAG = "RightDrag"**
- **String DOUBLE_TAP = "DoubleTap"**
- **String LONG_PRESS = "LongPress"**
- **String EXTREME_ZOOM = "ExtremeZoom"**
- **String ROTATION_ANTI= "RotationAnti"**
- **String EXTREME_ZOOM_IN = "ExtremeZoomIn"**
- **String EXTREME_ZOOM_OUT = "ExtremeZoomOut"**
- **String ROTATION_CLOCKWISE= "RotationClockwise"**

Once you follow all the above mentioned steps your application will start receiving message from our library and you can respond the way your want.