Experiment No:2

To create a "Hello World" application in Android using code, you'll need to follow these steps:

1. Set up Android Studio:
   - Download and install Android Studio from the official website.
   - Open Android Studio and create a new project.

2. Create a new Android project:
   - Choose "Empty Activity" as the template for your project.
   - Enter the project name, package name, and other details as required.
   - Click "Finish" to create the project.

3. Design the layout:
   - Open the `activity_main.xml` file in the `res/layout` directory.
   - Add a TextView element to display the "Hello World" message.
   - Your `activity_main.xml` file should look like this:

```xml
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/helloTextView"
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:textSize="24sp"
    android:layout_centerInParent="true" />
</RelativeLayout>
```

4. Modify the MainActivity code:

   - Open the `MainActivity.java` file in the `java/<your_package_name>` directory.

   - Find the `onCreate` method and add code to set the content view to your `activity_main.xml` layout.

```java
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
}
```

5. Run the app:

   - Connect your Android device to your computer or use an emulator.

   - Click the green "Run" button in Android Studio to build and run your app.

   - Select your device/emulator and click "OK" to install and launch the app.

6. Verify the output:

- Once the app is running on your device/emulator, you should see the "Hello World" message displayed on the screen

Code Explanation:

The XML code you provided is the layout file (`activity_main.xml`) for a "Hello World" Android application. Let's break down the key components:

1. `<?xml version="1.0" encoding="utf-8"?>`: This line specifies the XML version and encoding used in the file.

2. `<RelativeLayout>`: This is the root element of the layout file. It's a layout manager that arranges its children views relative to each other or to the parent.

3. `xmlns:android="http://schemas.android.com/apk/res/android"`: This XML namespace declaration is required for using Android-specific attributes and elements.

4. `xmlns:tools="http://schemas.android.com/tools"`: This XML namespace declaration is for design-time attributes provided by the Android Studio layout editor.

5. `android:layout_width="match_parent"` and `android:layout_height="match_parent"`: These attributes set the width and height of the RelativeLayout to match the parent's width and height, occupying the entire screen.

6. `tools:context=".MainActivity"`: This is a design-time attribute used by Android Studio to associate the layout with the MainActivity class.

7. `<TextView>`: This is a TextView element, which displays text on the screen.

8. `android:id="@+id/helloTextView"`: This assigns an ID to the TextView, which can be used to reference this view in the Java code.

9. `android:layout_width="wrap_content"` and `android:layout_height="wrap_content"`: These attributes set the width and height of the TextView to wrap its content.

10. `android:text="Hello World!"`: This sets the text displayed by the TextView to "Hello World!"

11. `android:textSize="24sp"`: This sets the text size to 24 scale-independent pixels (sp).

12. `android:layout_centerInParent="true"`: This centers the TextView horizontally and vertically within the parent RelativeLayout.

The code snippet you provided is from the `MainActivity.java` file in an Android project. Let's break down what each line does:

1. `@Override`: This annotation indicates that the `onCreate` method overrides a method from a superclass or interface.

2. `protected void onCreate(Bundle savedInstanceState) {`: This line declares the `onCreate` method, which is a lifecycle method called when the activity is created. It takes a `Bundle` parameter named `savedInstanceState`, which is used to restore the activity's previous state if needed.

3. `super.onCreate(savedInstanceState);`: This line calls the superclass's `onCreate` method to perform any necessary initialization for the activity.

4. `setContentView(R.layout.activity_main);`: This line sets the content view of the activity to the layout defined in the `activity_main.xml` file. It inflates the XML layout and displays it on the screen when the activity is created.

Overall, this `onCreate` method is crucial in the Android activity lifecycle as it sets up the activity's user interface and initializes any necessary components.