BÀI 4. PHIẾU BÀI TẬP

BÀI TẬP TRƯỚC KHI LÊN LỚP: Bài 4.1 đến 4.6

BÀI TẬP THỰC HIỆN TRÊN LỚP: Bài 4.7, 4.8, 4.9

BÀI TẬP SAU KHI LÊN LỚP: Bài 4.10 đến 4.13

MỤC LỤC

Bài tập 4.1. Tính tổng 2 số .

Cho giao diện mẫu sau. Yêu cầu nhập vào giá trị 2 số a và b. Thực hiện tính tổng 2/Hiệu số và hiển thị kết quả tương tứng.

-Thiết kế giao diện có thể thay đổi về bố trí và sắp xếp, màu sắc, kích thước đảm bảo tính hài hoà trong hiển thị.

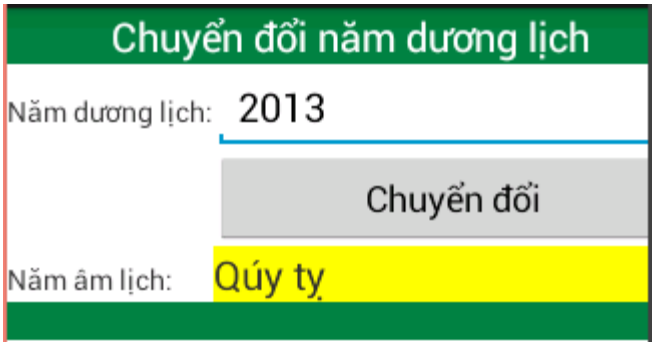
- Áp dụng các kiểu lập trình sự kiện đã học thực hiện tính tổng và hiệu hai số nhập vào qua ô EditText số a, số b.

Hướng dẫn thực hiện: Đề cương bài giảng mục 1.

Bài tập 4.2. Chuyển đổi năm âm lịch sang năm dương lịch.

Cho giao diện sau. Màn hình gồm:

- 1 ô EditText nhập năm dương lịch
- 1 TextView hiển thị kết quả chuyển đổi năm dương lịch sang năm âm lịch
- 1 Button chuyển đổi



Sử dụng một trong cách lập trình sự kiện đã học đổi năm dương lịch sang năm âm lịch. Khi người sử dụng nhập vào EditText giá trị là 1 năm Dương Lịch bất kỳ nào đó rồi nhấn nút “Chuyển đổi”, chương trình sẽ chuyển năm dương lịch thành năm âm lịch. Trong ví dụ trên nếu người sử dụng nhập 2022 thì sẽ ra năm âm lịch là “Nhâm Dần”.

Gợi ý thuật toán: Năm Âm lịch = Can + Chi,

Bảng Can: Can = Năm dương %10. Nếu số dư là 0-9 thì Can nhận các giá trị sau:

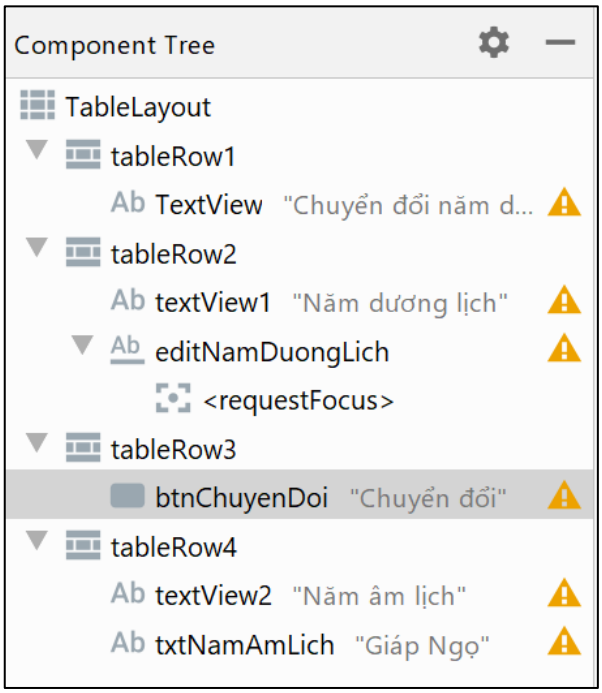
0	1	2	3	4	5	6	7	8	9
Canh	Tân	Nhân	Quý	Giáp	Ất	Bính	Đinh	Mậu	Kỷ

Bảng Chi: Chi = Năm Dương%12. Nếu số dư từ 0- 11 thì Chi nhận các giá trị sau

0	1	2	3	4	5	6	7	8	9	10	11
Thân	Dậu	Tuất	Hợi	Tý	Sửu	Dần	Mão	Thìn	Tỵ	Ngọ	Mùi

Can = Năm dương %10

Hướng dẫn thực hiện



Hướng dẫn thực hiện: Mã lệnh xử lý

MainActivity.java

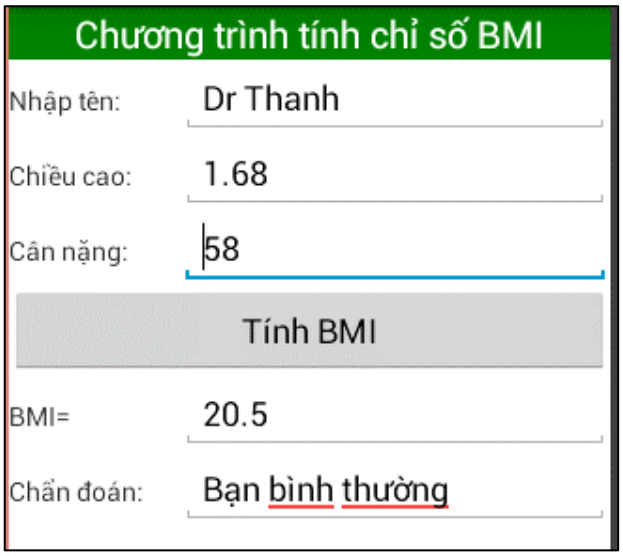
```
public class MainActivity extends AppCompatActivity {
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button btnChuyenDoi=(Button)findViewById(R.id.btnChuyenDoi);
        btnChuyenDoi.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
        //bước 1: lấy giá trị nhập vào từ editTexxt
        EditText editNamDuongLich = ???
         //ép giá trị nhập về vào số nguyên
         int namDuongLich=???
```

```
        //bước 3: Có số nguyên là đại diện cho năm dương
        //áp dụng hướng dẫn tính Can+chi
        int can=namDuongLich % 10,chi=namDuongLich % 12;
        String tinhCan="",tinhChi="";
        switch (can){
               case 0: tinhCan="Canh";break;
                case 1: tinhCan="Tân";break;
                case 2: tinhCan="Nhâm";break;
                case 3: tinhCan="Quý";break;
                case 4: tinhCan="Giáp";break;
                case 5: tinhCan="Ất";break;
                case 6: tinhCan="Bính";break;
                case 7: tinhCan="Đinh";break;
                case 8: tinhCan="Mậu";break;
                case 9: tinhCan="Kỷ";break;
         }
         switch (chi){                  {
               case 0: tinhChi="Thân";break;
                case 1: tinhChi="Dậu";break;
                case 2: tinhChi="Tuất";break;
                case 3: tinhChi="Hợi";break;
                      case 4: tinhChi="Tý";break;
                      case 5: tinhChi="Sửu";break;
                      case 6: tinhChi="Dần";break;
                      case 7: tinhChi="Mão";break;
                      case 8: tinhChi="Thìn";break;
                      case 9: tinhChi="Tỵ";break;
                      case 10: tinhChi="Ngọ";break;
                      case 11: tinhChi="Mùi";break;
                  }
                  //gán giá trị cho textNam Âm lịch
                  TextView
txtNamAmLich=(TextView)findViewById(R.id.txtNamAmLich);
                  txtNamAmLich.setText((tinhCan+tinhChi)+"");

              }
         });
    }
}
```

Bài tập 4.3. Bài tập tính chỉ số cơ thể BMI

Cho giao diện sau. Màn hình gồm:

- Các EditText nhập tên, chiều cao, cân nặng. 2 EditText nhận kết quả chỉ số BMI và chuẩn đoạn tình trạng cơ thể
- 1 Button tính BMI



Viết chương trình tính chỉ số cơ thể BMI (Body Mass Index)

Công thức chỉ số cơ thể tính như sau

Gói W là cân nặng (tính bằng kg).

H là chiều cao (tính bằng mét)

BMI= W/(H*H)

BMI <18: Người gầy

BMI: 18 đến 24.9 : Người bình thường

BMI: 25 đến 29.9 : Người béo phì độ 1

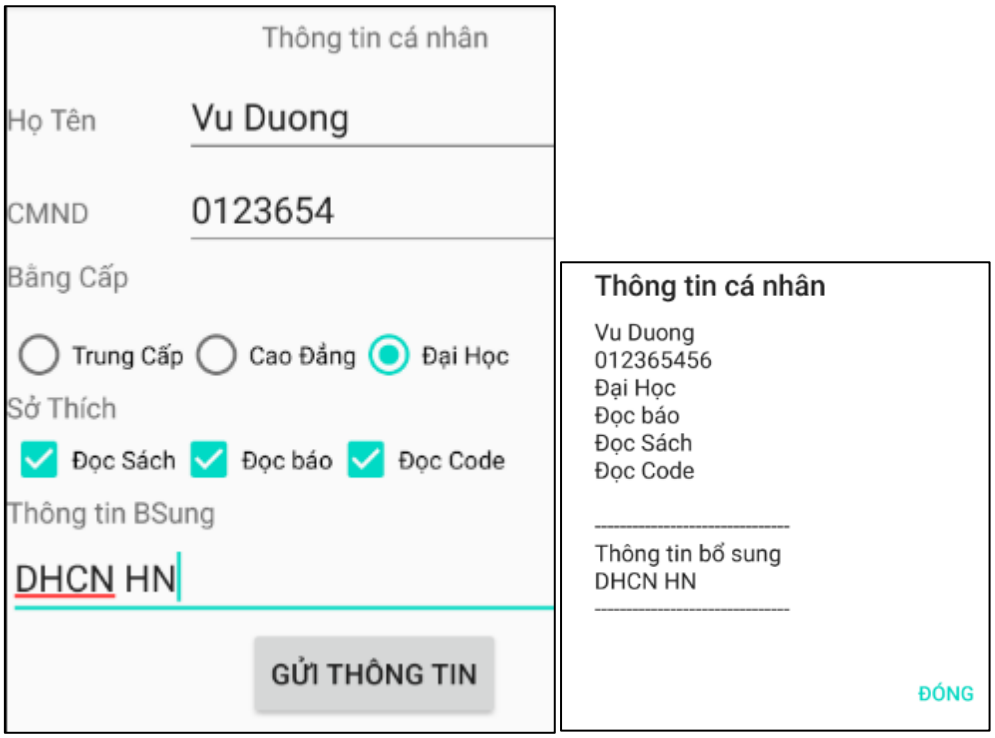
BMI: 30 đến 34.9 : Người béo phì độ 2

BMI: >35 : Người béo phì độ 3

Bài tập 4.4. Bài tập viết phần mềm xử lý thông tin cá nhân.

Cho màn hình sau. Màn hình bao gồm:

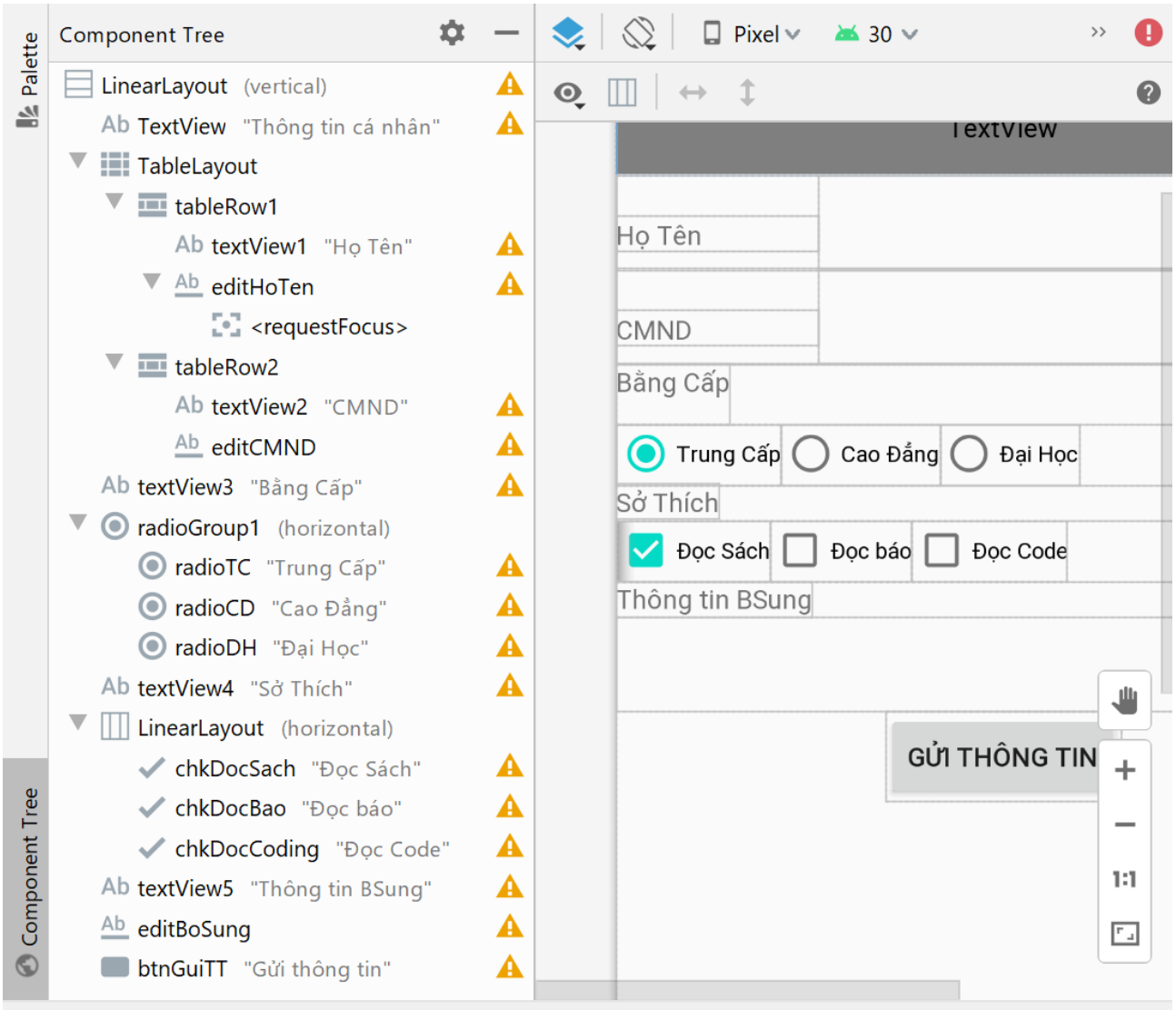
- 2 EditText nhập họ tên và CMND
- 3 RadioButton lựa chọn Bằng cấp
- 3 CheckBox cho phép lựa chọn sở thích
- 1 EditText cho phép nhập thông tin bổ sung
- 1 Button gửi thông tin



Xử lý nút Gửi thông tin. Kiểm tra bằng cấp, sở thích và thông tin bổ sung sau hiển thị thông tin cá nhân qua Diaglog

Hướng dẫn thực hiện

1. Thiết kế giao diện giả sử có mẫu như sau



MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    //khai báo các control
    EditText editTen,editCMND,editBoSung;
    CheckBox chkDocBao,chkDocSach,chkDocCode;
    Button btnGuiTT;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getWidget();
        btnGuiTT.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                doShowInformation();
```

```
            }
        });
    }
    public void getWidget(){
        editTen=(EditText)findViewById(R.id.editHoTen);
        editCMND=(EditText)findViewById(R.id.editCMND);
        editBoSung=(EditText)findViewById(R.id.editBoSung);
        chkDocBao=(CheckBox)findViewById(R.id.chkDocBao);
        chkDocSach=(CheckBox)findViewById(R.id.chkDocSach);

chkDocCode=(CheckBox)findViewById(R.id.chkDocCoding);
        btnGuiTT=(Button)findViewById(R.id.btnGuiTT);
    }
    public void doShowInformation() {
        //kiểm tra tên hợp lệ
        String ten=editTen.getText()+"";
        ten=ten.trim();
        if (ten.length()<3) {
            editTen.requestFocus();
            editTen.selectAll();
            Toast.makeText(this, "Tên phải lớn hơn 3 ký tự",
Toast.LENGTH_LONG).show();
            return;
        }
        //kiểm tra chứng minh thư hợp lệ
        String cmnd=editCMND.getText()+"";
        cmnd=cmnd.trim();
        if (cmnd.length()!=9) {
            editCMND.requestFocus();
            editCMND.selectAll();
            Toast.makeText(this, "CMND phải đugns 9 ký tự",
Toast.LENGTH_LONG).show();
            return;
        }
        //kiểm tra bằng cấp
        String bangCap="";
        RadioGroup
group=(RadioGroup)findViewById(R.id.radioGroup1);
        int id=group.getCheckedRadioButtonId();
        if (id==-1){
            Toast.makeText(this, "Phải chọn bằng
cấp",Toast.LENGTH_LONG).show();
            return;
        }
        RadioButton rad=(RadioButton)findViewById(id);
        bangCap=rad.getText()+"";
        //kiểm tra sở thích
        String soThich="";
        if (chkDocBao.isChecked())
```

```
soThich+=chkDocBao.getText() +"\n";
        if (chkDocSach.isChecked())
soThich+=chkDocSach.getText() +"\n";
        if (chkDocCode.isChecked())
soThich+=chkDocCode.getText() +"\n";
        //lấy thông tin bổ sung
        String boSung=editBoSung.getText()+"";
        AlertDialog.Builder builder=new
AlertDialog.Builder(this);
        builder.setTitle("Thông tin cá nhân");
        builder.setPositiveButton("Đóng", new
DialogInterface.OnClickListener() {

            @Override
            public void onClick(DialogInterface dialog, int
which) {
                // TODO Auto-generated method stub
                dialog.cancel();
            }
        }); //end of setPossitiveButton
        //tạo nội dung cho builder
        String msg=ten+"\n";
        msg+=cmnd+"\n";
        msg+=bangCap+"\n";
        msg+=soThich+"\n";
        msg+="--------------------------------\n";
        msg+="Thông tin bổ sung\n";
        msg+=boSung+"\n";
        msg+="--------------------------------\n";
        builder.setMessage(msg);//thiết lập mội dung cho
mess
        builder.create().show();
    }
}
```

Bài tập 4.5. ListView đơn giản

Giả sử có giao diện đơn giản hiển thị tên các thành phố lớn như sau:

ListView chứa danh sách các thành phố. Khi chọn 1 dòng trong ListView thì vị trí và giá trị hiển trở lại TextBox có nền xanh trên đỉnh màn hình.

Hướng dẫn thực hiện: Đề cương bài giảng mục 2 phần tương tác với ListView

Bài tập 4.6. Luyện tập ListView và bài toán quản lý nhân viên

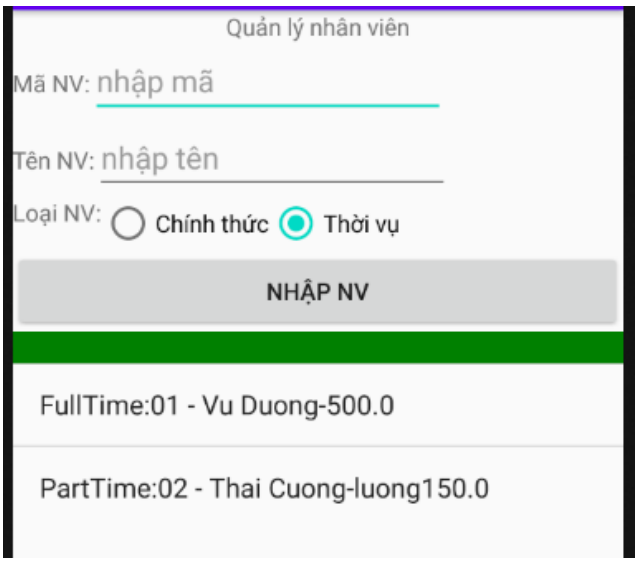
Cho bài toán quản lý nhân viên: Nhân viên có 2 loại: Nhân viên chính thức (EmployeeFullTime) và nhân viên thời vụ (EmployeePartime). Các nhân viên mô tả thông qua mã nhân viên (id) và tên nhân viên (name).

- Mỗi nhân viên sẽ có cách tính lương khác nhau (tên phương thức tính lương giống nhau). Đối với FullTime thì lương 500, EmployeePartime lương 150.
- Mỗi nhân viên có phương thức toString để xuất thông tin. Nội dung xuất khác nhau: thêm FullTime đằng sau Id và Name đối với nhân viên chính thức. Thêm Partime đằng sau Id và Name đối với nhân viên thời vụ.

Màn hình giao diện như sau:

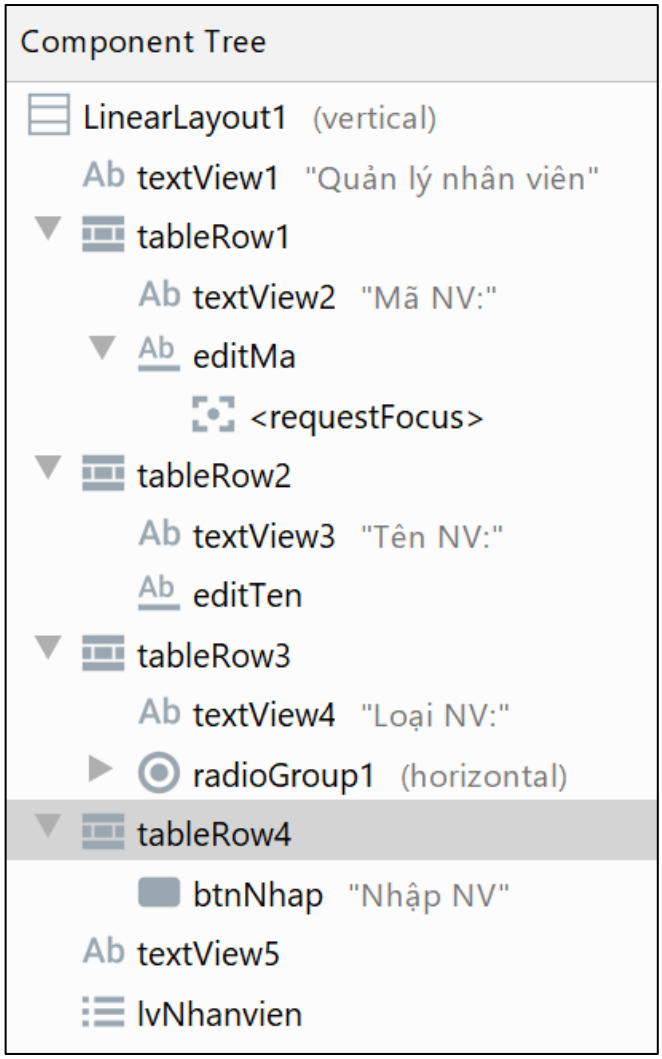
Màn hình bao gồm:

- 2 ô EditText nhập liệu tên đăng nhập và mật khẩu.
- 1 Checkbox lưu thông tin
- 2 Button nhập và thoát



Gợi ý thực hiện:

- Cây thiết kế giao diện:



- Activity_main.xml: sinh viên tự sinh mã và xây dựng giao diện có đặt tên theo gợi ý
- Xây dựng lớp cơ sở: Nhân viên (Employee) lớp cha và 2 lớp con kế thừa: Nhân viên full time (EmployeefullTime) và Nhân viên thời vụ (EmployeePartime)

Employee.java.

```
package vuduong.cpm;
public abstract class Employee {
   private String id;
   private String name;
   public abstract double tinhLuong();
   public String getId() {
      return id;
   }
   public String getName() {
      return name;
   }
   public void setId(String id) {
      this.id = id;
   }
   public void setName(String name) {
      this.name = name;
   }
   @Override
   public String toString() {
      return this.id+" - "+this.name;
   }
}
```

EmployeeFullTime.java

```
public class EmployeeFullTime extends Employee{
     @Override
     public String toString() {
        return "FullTime:"+ super.toString() +"-
"+tinhLuong();
      }
    @Override
    public double tinhLuong() {
       return 500;
    }
}
```

EmployeePartTime.java

```
public class EmployeePartTime extends Employee{
   @Override
    public double tinhLuong() {
       return 150;
  }
   @Override
   public String toString() {
     return "PartTime:"+ super.toString() +"-luong"+
tinhLuong();  }}
```

ActivityMain.java

Sinh viên hoàn thiện nội dung mã hóa theo gợi ý. ??? là các phần thông tin còn thiếu.

```
public class MainActivity extends AppCompatActivity {
    EditText editId,editName;
     Button btnNhap;
     RadioGroup radGroup;
     ListView lvNhanvien;
     ArrayList<Employee>arrEmployee=new
ArrayList<Employee>();
     ArrayAdapter<Employee>adapter=null;
     //Khai báo 1 employee object.
     Employee employee=null;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //kết nối các điều khiển thông qua findByViewId
        editId=(EditText) findViewById(???);
        editName=(EditText) findViewById(???);
        btnNhap=(Button) findViewById((???););
        radGroup=(RadioGroup) findViewById((???););
        lvNhanvien=(ListView) findViewById((???););
         //đưa Data Source là các employee vào Adapter
        adapter=new ArrayAdapter<Employee>((???));
         //đưa adapter vào ListView
        lvNhanvien.setAdapter(adapter);
        //Xử lý thông tin trong sự kiện nút lệnh nhập
        btnNhap.setOnClickListener(???);
        //Xử lý thông tin khi chọn donngf trong listView
        lvNhanvien.setOnItemClickListener(???);
     }
}
```

Bài tập 4.7. Tùy biến ListView bài 4.5

Tùy biến lại listView trong bài 4.5 thành giao diện có mô tả như sau

Hướng dẫn thực hiện: Đề cương bài giảng mục 2.2.

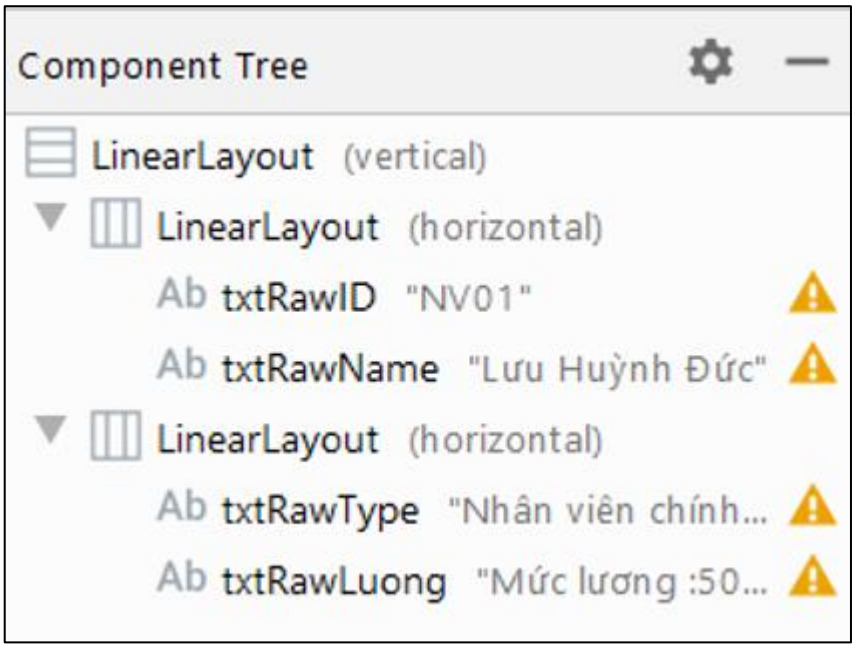
Bài tập 4.8. Tùy biến ListView Quản lý nhân viên

Tùy biến hiển thị listView trong bài 4.4 có giao diện như sau:

Hướng dẫn thực hiện:

- Thiết kế giao diện mẫu cho 1 dòng thay thế:

- **myItemList.java** giao diện hiển thị có cấu trúc cây như sau:

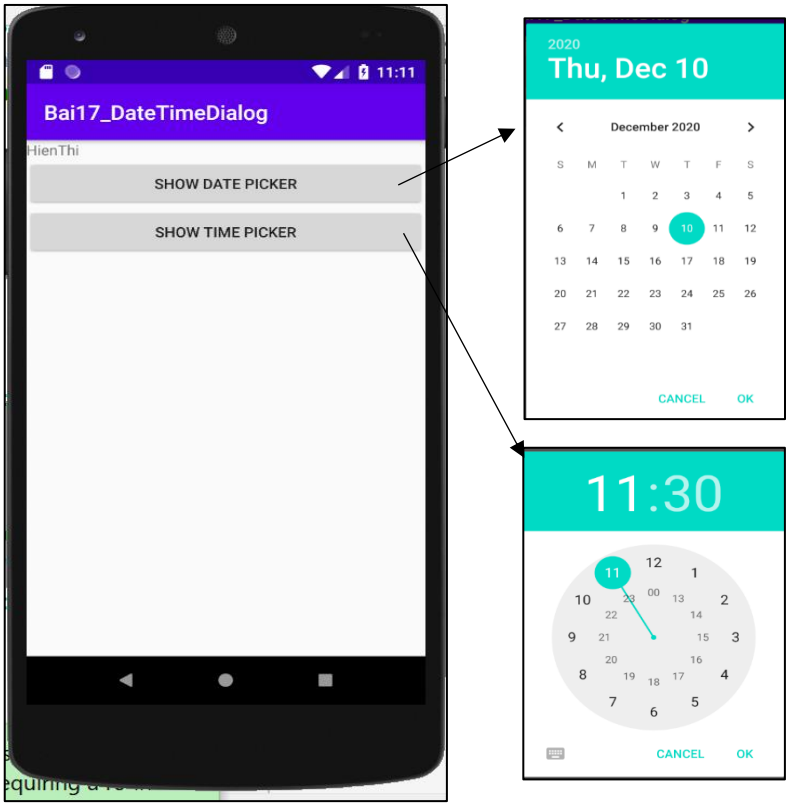


- Mã hóa tùy biến: Sinh viên tự triển khai.

Bài tập 4.9. Demo xử lý thông tin với DatePicker và TimePicker

Cho giao diện màn hình chính như sau: Màn hình gồm 2 nút lệnh. Nút lệnh ShowDatePiker khi được chọn sẽ gọi hiển thị của sổ chọn ngày. Khi ngày được chọn và đóng của sổ ngày, thông tin sẽ hiển thị trở lại màn hình chính với ngày đã chọn

Nút lện ShowTimePicker được chọn sẽ gọi cửa sổ hiển thị giờ. Khi chọn giở hiển thị và đóng cửa sổ giờ, thông tin cg hiển trở lại mành hình chính.



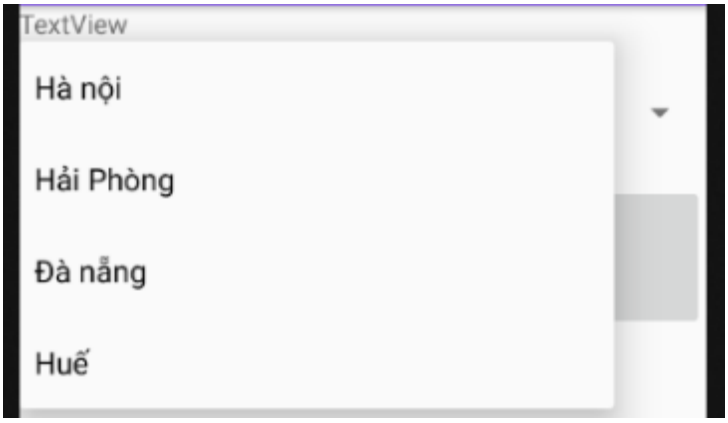
Trải nghiệm ứng dụng:

- Chọn ShowDatePicker: cửa sổ Datepicker hiển thị. Khi chọn ShowTimePicker cửa số TimePicker hiển thị.
- Trong khi các cửa sổ hiển thị do người dùng chọn date hay time; chọn ok. Thông tin vừa chọn hiển thị trong text minh họa.

Hướng dẫn thực hiện: Mục 4 đề cương bài giảng

Bài tập 4.10. Spinner lưu trữ danh sách các tỉnh thành phố

Cho màn hình thực thi có giao diện như sau



Spinner nhận dữ liệu là danh sách các thành phố. Chọn 1 dòng trong Spinner thì dữ liệu dòng cọn hiển thị lên TextView.

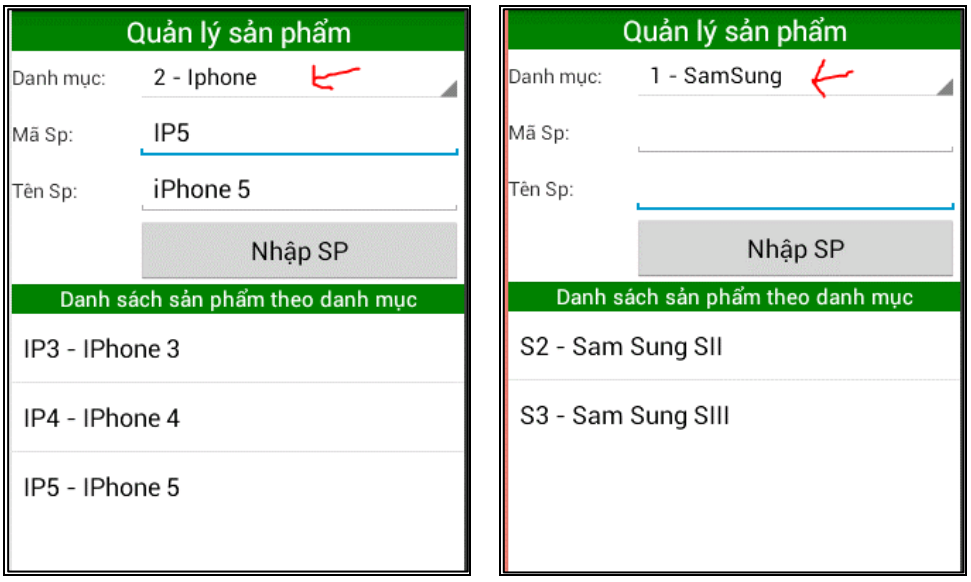
Hướng dẫn thực hiện: Bài giảng điện tử mục 3.

Bài tập 4.11. Quản lý sản phẩm

Quản lý sản phẩm. Cho mẫu giao diện quản lý sản phẩm như hình vẽ sau. Danh mục sản phẩm là spinner lưu trữ tên danh mục sản phẩm. Viết chương trình thực hiện chọn danh mục sản phẩm, danh sách sản phẩm theo danh mục được lọc theo. Chọn Nhập SP thì thông tin sản phẩm được nhập vào ListView phía dưới.

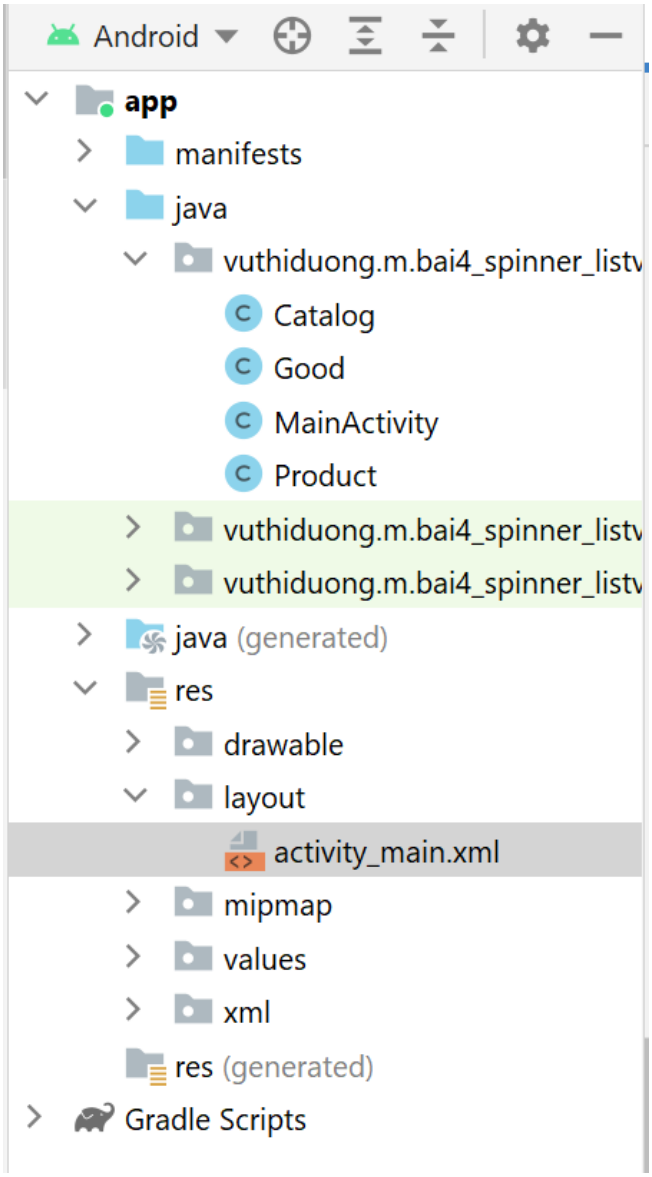
Màn hình bao gồm:

- 1 Spinner lưu trữ danh mục sản phẩm
- 2 ô EditText nhập liệu mã sản phẩm và tên sản phẩm
- 1 Button nhập sản phẩm

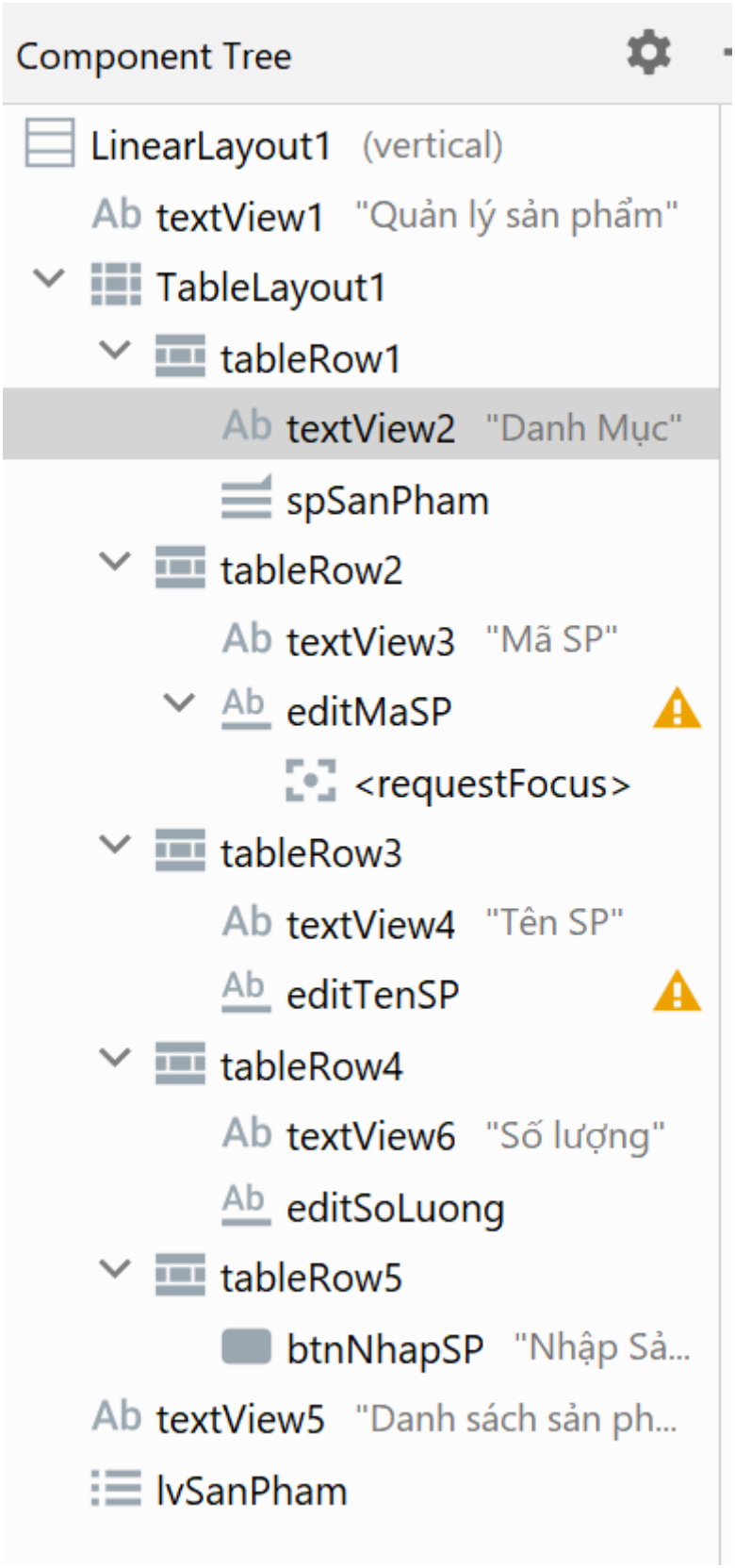


Hướng dẫn thực hiện:

1. Cấu trúc Project như sau



2. Thiết kế giao diện Sinh viên tự thiết kế theo cây mô tả sau



Lớp Hàng hóa mô tả thông qua lớp Good - lớp cha như sau

Good.java

```
package vuthiduong.m.bai4_spinner_listview_v2;

public class Good {
    public String maSP;
    public String tenSP;
    //hàm tạo
    public Good(String maSP, String tenSP) {
        super();
        this.maSP = maSP;
        this.tenSP = tenSP;
    }
    public Good(){}
    //các hàm getter/setter
    public String getMaSP() {
        return maSP;
    }
    public void setMaSP(String maSP) {
        this.maSP = maSP;
    }
    public String getTenSP() {
        return tenSP;
    }
    public void setTenSP(String tenSP) {
        this.tenSP = tenSP;
    }
    //ghi đề tostring
    @Override
    public String toString() {
        return "ma SP=" + maSP + "\t tenSP=" + tenSP ;
    }
    //chèn equal &hashCode
    @Override
    public int hashCode() {
        final int prime = 31;
        int result = 1;
        result = prime * result + ((maSP == null) ? 0 :
maSP.hashCode());
        return result;
    }
    @Override
    public boolean equals(Object obj) {
        if (this == obj)
            return true;
        if (obj == null)
            return false;
        if (getClass() != obj.getClass())
```

```
            return false;
        Good other = (Good) obj;
        if (maSP == null) {
            if (other.maSP != null)
                return false;
        } else if (!maSP.equals(other.maSP))
            return false;
        return true;
    }
}
```

Sản phẩm kế thừa từ hàng hóa và có thêm số lương.

Product.java

```
package vuthiduong.m.bai4_spinner_listview_v2;
//luu thông tin sản phẩm.
public class Product extends Good{
    private int soLuong;
    public Product() {
        // TODO Auto-generated constructor stub
        super();
        soLuong=0;
    }
    public Product(String maSP, String tenSP, int
soLuong) {
        super(maSP, tenSP);
        this.soLuong = soLuong;
    }
    @Override
    public String toString() {
        return  super.toString() + " \t số
lượng="+soLuong;
    }
    public int getSoLuong() {
        return soLuong;
    }
    public void setSoLuong(int soLuong) {
        this.soLuong = soLuong;
    }
}
```

Danh mục hàng hóa. Mỗi danh mục gồm nhiều hàng:

Catalog.java

```
import java.util.ArrayList;
//lớp chứa danh sách sản phẩm
public class Catalog {
   private String maDM,tenDM;
   private ArrayList<Product> dsSanPham =null;
   public Catalog(String ma, String ten) {
      // TODO Auto-generated constructor stub
      maDM=ma;tenDM=ten;
      dsSanPham = new ArrayList<Product>();
   }
   /*
    * Kiểm tra sản phẩm đã tồn tại trong danh mục chưa
    * @para p - Product
    * @return true nếu tồn tại
    */
   public boolean kiemTraSanPham(Product p){
      for(Product p1:dsSanPham){
         if
(p1.getMaSP().trim().equalsIgnoreCase(p.getMaSP().trim
()))
            return true;
      }
      return false;
   }
   /*
    * Thêm 1 sản phẩm vào danh mục
    * thêm thành công true
    */
   public boolean addSP(Product p)    {
      boolean kiemTra=kiemTraSanPham(p);
      if (!kiemTra)
      {
         dsSanPham.add(p);
         return true;
      }
      else return false;
   }
   public ArrayList<Product> getDsSanPham() {
         return dsSanPham;
   }
   @Override
   public String toString() {
      return maDM + "-" + tenDM ;
   }
}
```

Các xử lý trong hàm main

MainActivity.java

```
public class MainActivity extends AppCompatActivity {

    Spinner spinDanhMuc;
    EditText editMaSP,editTenSP,editSoLuong;
    Button btnNhap;
    ListView lvSanPham;
    //khai báo cặp đối tượng dùng cho Spineer
    ArrayList<Catalog> arraySpinner=new
ArrayList<Catalog>();
    ArrayAdapter<Catalog> adapterSpinner =null;

    //khai báo cặp đối tượng dùng cho listView
    ArrayList<Product> arrayListView=new
ArrayList<Product>();
    ArrayAdapter<Product> adapterListView=null;

    @Override
    protected void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //lấy các control gắn với Activity
        getWidgetsControl();
//        //Giả dữ liệu mặc định
        fakeDataCatalog();
        addEventsFormWidGets();
    }
    private void getWidgetsControl(){
      //Sinh viên tự làm lấy các điều khiển giao diện
tương ứng
      //cấu hình cho Spinner. Điền cụ thể phần …
      adapterSpinner=;
      adapterSpinner.setDropDownViewResource(….);
      spinDanhMuc.setAdapter(adapterSpinner);
      //cấu hình cho listView
      adapterListView=new ArrayAdapter<Product>(...);
        lvSanPham.setAdapter(adapterListView);
    }
    //hàm giả dữ liệu tạo 4 danh mục mặ định cho
Spinner
    private void fakeDataCatalog()   {
        Catalog cat1=new Catalog("1","SamSung");
        Catalog cat2=new Catalog("2","Nokia");
        Catalog cat3=new Catalog("3","IPAD");
        Catalog cat4=new Catalog("4","HTC");

        arraySpinner.add(cat1);
```

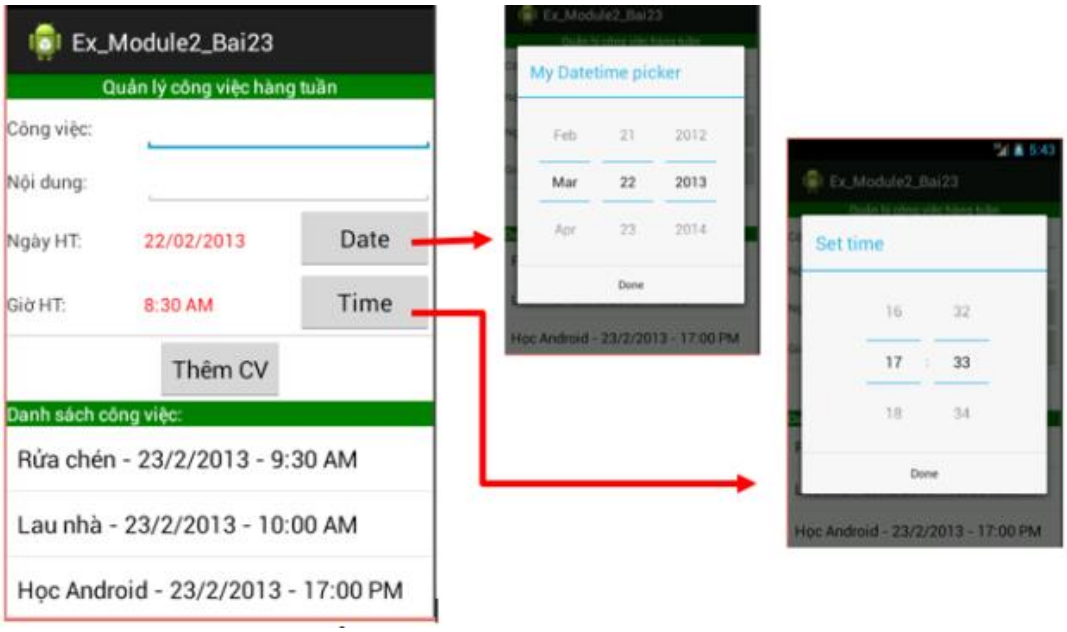
```
            arraySpinner.add(cat2);
            arraySpinner.add(cat3);
            arraySpinner.add(cat4);

            adapterSpinner.notifyDataSetChanged();
        }
        /*
         * Hàm gán sự kiện cho Button và spinner
         *
         */
        private void addEventsFormWidGets() {
          //Sinh viên tự hoàn thiện
}
```

Bài tập 4.12. Quản lý công việc với DatePicker và TimePicker.

Thực hiện thiết kế và viết chương trình cho màn hình có mẫu như hình vẽ sau để quản lý công việc cá nhân hàng tuần. Trong đó công việc, nội dung là các ô nhập liệu. Để nhập dữ liệu cho 2 ô ngày và giờ thì thực hiện gọi các điều khiển TimePicker, DatePicker.

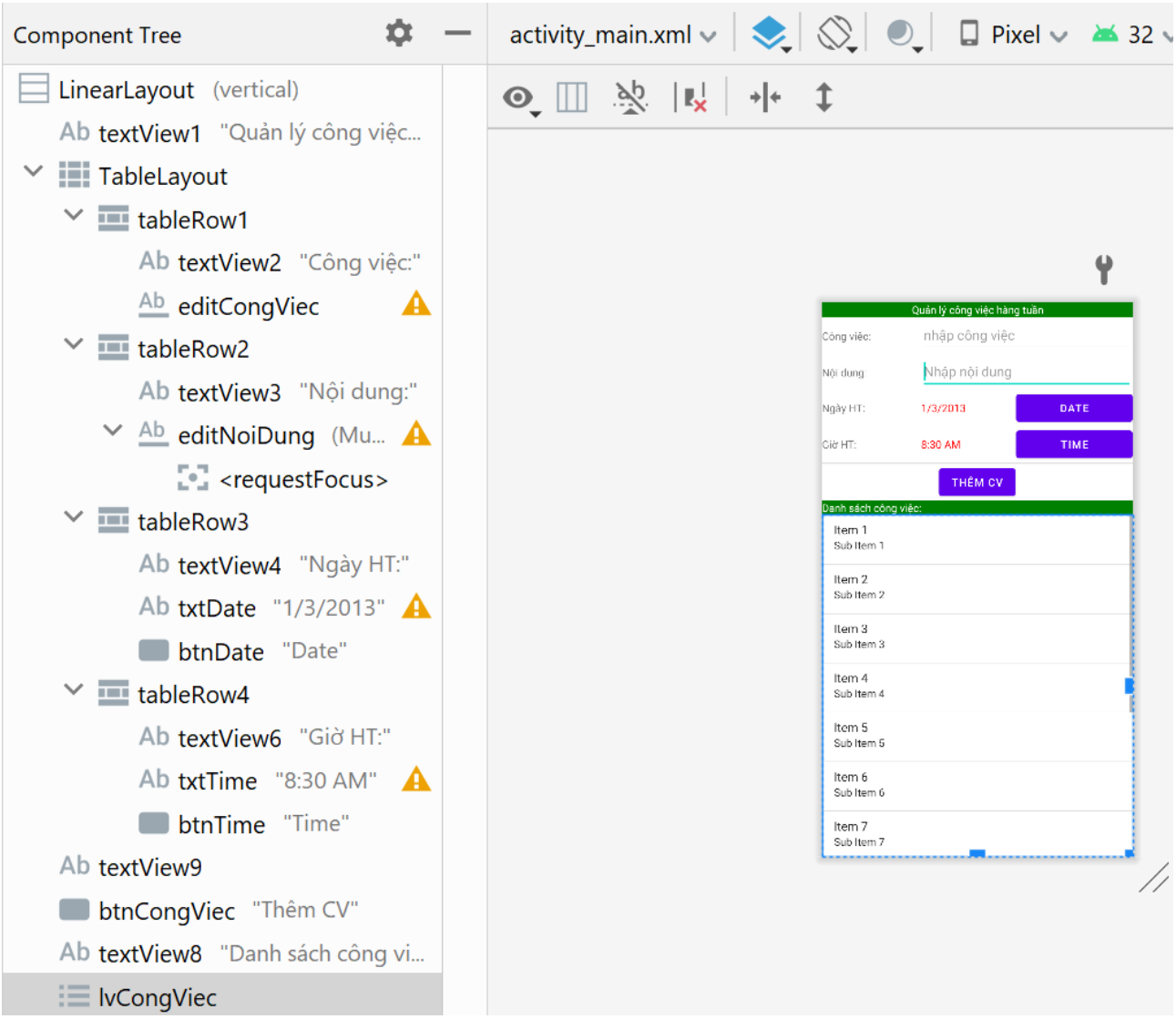
Màn hình bao gồm:

- 2 ô EditText nhập liệu công việc và nội dung công việc
- 2 EditText nhập kết quả ngày và giờ làm việc. Kế tiếp là 2 nút lệnh Date và Time
- 1 nút lệnh Thêm CV để thêm công việc
- 1 TextView có nền màu xanh, chữ trắng hiển thị nội dung danh sách công việc
- 1 ListView hiển thị nội dung công việc.



Hướng dẫn thực hiện:

1. Thiết kế giao diện



2. Thiết kế lớp cơ sở mô tả công việc

JobInWeek.java

```
class JobInWeek {
    private String title;
    private String desciption;
    private Date dateFinish;
    private Date hourFinish;
    public JobInWeek(String title, String desciption,
Date dateFinish, Date hourFinish) {
        this.title = title;
        this.desciption = desciption;
        this.dateFinish = dateFinish;
        this.hourFinish = hourFinish;
    }

    public JobInWeek() {

    }
```

```
    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }

    public String getDesciption() {
        return desciption;
    }

    public void setDesciption(String desciption) {
        this.desciption = desciption;
    }

    public Date getDateFinish() {
        return dateFinish;
    }

    public void setDateFinish(Date dateFinish) {
        this.dateFinish = dateFinish;
    }

    public Date getHourFinish() {
        return hourFinish;
    }

    public void setHourFinish(Date hourFinish) {
        this.hourFinish = hourFinish;
    }
    public String getDateFormat(Date d)
    {
        SimpleDateFormat dft=new
SimpleDateFormat("dd/MM/yyyy", Locale.getDefault());
        return dft.format(d);
    }
    /**
     * lấy định dạng giờ phút
     * @param d
     * @return
     */
    public String getHourFormat(Date d)
    {
        SimpleDateFormat dft=new SimpleDateFormat("hh:mm
a", Locale.getDefault());
        return dft.format(d);
    }
    @Override
```

```
    public String toString() {
        return this.title+"-"+
                getDateFormat(this.dateFinish)+"-"+
                getHourFormat(this.hourFinish);
    }
}
```

3. MainActivity.java

```
package vuthiduong.m.bai4_12_todolist;

public class MainActivitySV extends AppCompatActivity {
    TextView txtDate,txtTime;
    EditText editCv,editNd;
    Button btnDate,btnTime,btnAdd;
    //Khai báo Datasource lưu trữ danh sách công việc
    ArrayList<JobInWeek> arrJob=new
ArrayList<JobInWeek>();
    //Khai báo ArrayAdapter cho ListView
    ArrayAdapter<JobInWeek> adapter=null;
    ListView lvCv;
    Calendar cal;
    Date dateFinish;
    Date hourFinish;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        getFormWidgets();
        getDefaultInfor();
        addEventFormWidgets();
    }
    /**
     * hàm dùng để load các control theo Id
     */
    private void getFormWidgets() {
        //SV viết lệnh lấy các điều khiển tương ứng
        //Gán DataSource vào ArrayAdapter
        adapter=new ArrayAdapter<JobInWeek>
                (this,
                android.R.layout.simple_list_item_1,
                arrJob);
        //gán Adapter vào ListView
        lvCv.setAdapter(adapter);
    }
    /**
     * Hàm lấy các thông số mặc định khi lần đầu tiền
chạy ứng dụng
```

```
     */
    private void getDefaultInfor() {
        //sv tự thực hiện
    }
    private void addEventFormWidgets() {
        btnDate.setOnClickListener(new MyButtonEvent());
        btnTime.setOnClickListener(new MyButtonEvent());
        btnAdd.setOnClickListener(new MyButtonEvent());
        lvCv.setOnItemClickListener(new
MyListViewEvent());
        lvCv.setOnItemLongClickListener(new
MyListViewEvent());
    }
    /**
     * Class sự kiện của các Button
     *
     */
    private class MyButtonEvent implements
            View.OnClickListener
    {
        @Override
        public void onClick(View v) {
            switch(v.getId())
            {
                case R.id.btnDate:
                    showDatePickerDialog();
                    break;
                case R.id.btnTime:
                    showTimePickerDialog();
                    break;
                case R.id.btnCongViec:
                    processAddJob();
                    break;
            }
        }
    }
/**
 * Class sự kiện của ListView
 */
    private class MyListViewEvent implements
        AdapterView.OnItemClickListener,
        AdapterView.OnItemLongClickListener
    {
        @Override
        public void onItemClick(AdapterView<?>
adapterView, View view, int i, long l) {
       //Hiển thị nội dung công việc tại vị trí thứ arg2
       Toast.makeText(MainActivitySV.this,
                arrJob.get(i).getDesciption(),
```

```
                Toast.LENGTH_LONG).show();
                //trở lại giao diện trên sv tự thực hiện
        }

        @Override
        public boolean onItemLongClick(AdapterView<?>
adapterView, View view, int i, long l) {
            //Xóa vị trí thứ arg2
            arrJob.remove(i);
            adapter.notifyDataSetChanged();
            return false;
        }
    }
    /**
     * Hàm hiển thị DatePicker dialog
     */
    public void showDatePickerDialog() {
        //SV tự thực hiện
    }
    /**
     * Hàm hiển thị TimePickerDialog
     */
    public void showTimePickerDialog()  {
        //SV tự thực hiện
    }
    /**
     * Hàm xử lý đưa công việc vào ListView khi nhấn nút
Thêm Công việc
     */
    public void processAddJob()    {
        String title=editCv.getText()+"";
        String description=editNd.getText()+"";
        JobInWeek job=new JobInWeek(title, description,
dateFinish, hourFinish);
        arrJob.add(job);
        adapter.notifyDataSetChanged();
        //sau khi cập nhật thì reset dữ liệu và cho
focus tới editCV
        editCv.setText("");
        editNd.setText("");
        editCv.requestFocus();
    }
}
```

Bài tập 4.13. Hiển thị danh sách sinh viên sử dụng RecyclerView.

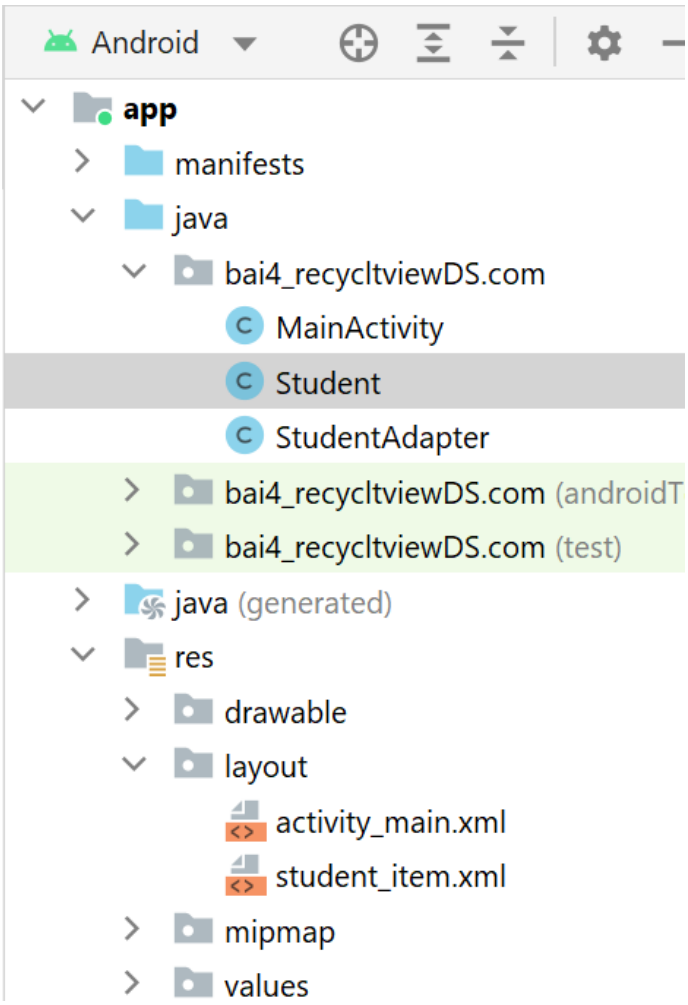
Sinh viên được mô tả thông qua họ tên, năm sinh. Xây dựng ứng dụng có mẫu như mô tả sau hiển thị danh sách sinh viên. Ứng dụng sử dụng RecyclerView lưu trữ danh sách sinh viên. Dữ liệu danh sách tự sinh 20 sinh viên. Ảnh minh họa cho các mẫu tự chọn.

Hình mô tả gợi ý giao diện như sau:

Danh sách sinh viên		
imageView	Họ và tên sinh viên	Nút lệnh chi tiết
imageView	Họ và tên sinh viên Năm sinh	Nút lệnh chi tiết

Hướng dẫn thực hiện:

Cấu trúc ứng dụng hoàn thành bài toán sẽ có hình vẽ mô tả gợi ý sau.



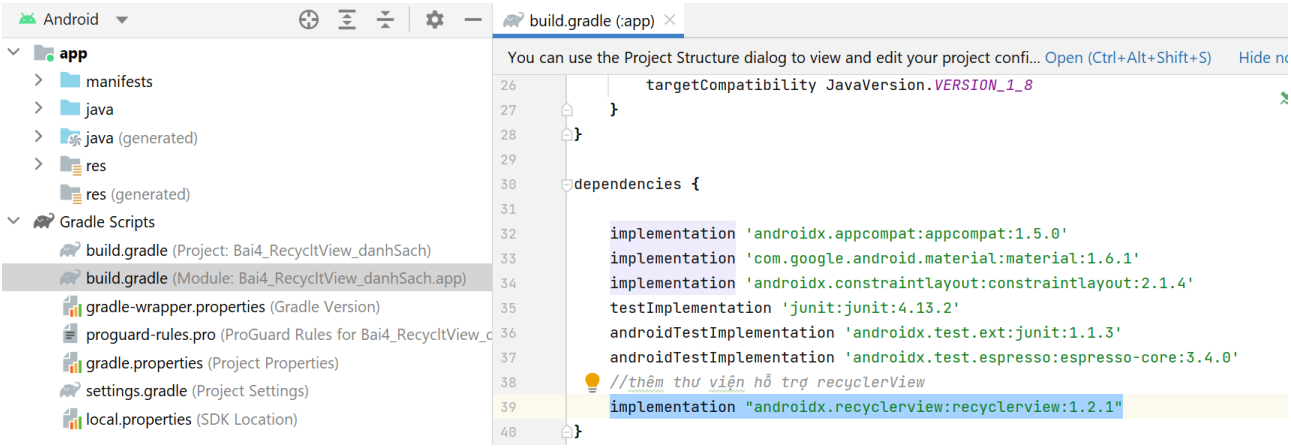
Chi tiết thực hiện theo 6 bước đã học:

Bước 1: tạo Project. Ví dụ đặt tên Bai4_RecyclerView_DS.

Thêm thư viện vào buile.grade (moduj) dòng lệnh:

```
implementation "androidx.recyclerview:recyclerview:1.2.1"
```

Minh họa hình vẽ sau:



Bước 2: Định nghĩa model class. Trong yêu cầu bài tập trên, tạo lớp Student có 2 thuộc tính: tên, và ngày sinh như sau

Student.java

```
public class Student {
    private String mName;
    private int birthYear;

    public void setmName(String mName) {
        this.mName = mName;
    }

    public void setBirthYear(int birthYear) {
        this.birthYear = birthYear;
    }
    public Student(String mName, int birthYear) {
        this.mName = mName;
        this.birthYear = birthYear;
    }
    public String getmName() {
        return mName;
    }

    public int getBirthYear() {
        return birthYear;
    }
}
```

Bước 3: Thiết kế giao diện ứng dụng chính

Activity_main.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/studentsList"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</LinearLayout>
```

Bước 4: Tạo Tạo một tệp XML để xác định hiển thị một item trong recyclerView

student_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <ImageView
        android:layout_width="60dp"
        android:layout_height="match_parent"
        android:src="@android:drawable/ic_menu_gallery"
        app:srcCompat="@android:drawable/btn_star_big_on"
        tools:srcCompat="@android:drawable/btn_star_big_on" />

    <LinearLayout
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:orientation="vertical"
        android:paddingLeft="5dp">

        <TextView
            android:id="@+id/studentname"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="dsafdsfdsfasf"
            android:textSize="16sp"
            android:textStyle="bold"
            />

        <TextView
            android:id="@+id/birthyear"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="dsfadsfdsf"
            android:textSize="14sp"
            android:textStyle="italic"
            />
    </LinearLayout>

    <Button
        android:id="@+id/detail_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingLeft="16dp"
        android:paddingRight="16dp"
        android:text="Chi tiết"
        android:textSize="10sp" />
</LinearLayout>
```

Bước 5: Tạo ra RecyclerView.Adapter và ViewHolder

Tạo khung 2 lớp lồng nhau:

```
public class StudentAdapter
        public class UserViewHolder extends RecyclerView.ViewHolder {
}
```

Hoàn thiện lớp UserViewHolder trước rồi tới lớp ngoài StudentAdapter. Chi tiết như sau:

```
public class UserViewHolder extends RecyclerView.ViewHolder {
    private View itemview;
```

```
    public TextView studentname;
    public TextView birthyear;
    public Button detail_button;

    public UserViewHolder(View itemView) {
        super(itemView);
        itemview = itemView;
        studentname = itemView.findViewById(R.id.studentname);
        birthyear = itemView.findViewById(R.id.birthyear);
        detail_button = itemView.findViewById(R.id.detail_button);

        //Xử lý khi nút Chi tiết được bấm
        detail_button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Toast.makeText(view.getContext(),
                                studentname.getText() +" - sinh năm: "
                                        + birthyear, Toast.LENGTH_SHORT)
                        .show();
            }
        });
    }
}
```

Lớp ngoài: Khung tổng quát StudentAdapter.java

```
public class StudentAdapter extends RecyclerView.Adapter<StudentAdapter.UserViewHolder>{
    //Dữ liệu hiện thị là danh sách sinh viên
    private List mStutents;
    // Lưu Context để dễ dàng truy cập
    private Context mContext;
    public StudentAdapter(List dsStudent, Context mContext) {...}
    @NonNull
    @Override
    public UserViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {...}

    //chuyển dữ liệu phần tử vào ViewHolder
    @Override
    public void onBindViewHolder(@NonNull UserViewHolder holder, int position) {...}
    //cho biết số phần tử của dữ liệu
    @Override
    public int getItemCount() {...}

    public class UserViewHolder extends RecyclerView.ViewHolder {...}
}
```

Nội dung đầy đủ StudentAdapter.java

```
public class StudentAdapter extends
RecyclerView.Adapter<StudentAdapter.UserViewHolder>{
    //Dữ liệu hiện thị là danh sách sinh viên
    private List mStutents;
    // Lưu Context để dễ dàng truy cập
    private Context mContext;
    public StudentAdapter(List dsStudent, Context mContext) {
        this.mStutents = dsStudent;
        this.mContext = mContext;
        notifyDataSetChanged();
    }
    @NonNull
```

```
    @Override
    public UserViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int
viewType) {
        Context context = parent.getContext();
        LayoutInflater inflater = LayoutInflater.from(context);
        // Nạp layout cho View biểu diễn phần tử sinh viên
        View studentView =
                inflater.inflate(R.layout.student_item, parent, false);

        UserViewHolder viewHolder = new UserViewHolder(studentView);
        return viewHolder;
    }

    //chuyển dữ liệu phần tử vào ViewHolder
    @Override
    public void onBindViewHolder(@NonNull UserViewHolder holder, int position)
{
        Student student = (Student) mStutents.get(position);
        if (student ==null)
            return;
        holder.studentname.setText(student.getmName());
        holder.birthyear.setText(student.getBirthYear()+"");
    }
    //cho biết số phần tử của dữ liệu
    @Override
    public int getItemCount() {
        if (mStutents ==null)
            return 0;
        else
            return mStutents.size();
    }
public class UserViewHolder extends RecyclerView.ViewHolder {
        //Đưa nội dung của lớp đã mô tả UserViewHolder như trên vào đây
}
}
```

Bước 6: Kết nối adapter với data source để đưa vào RecyclerView.

MainActivity.java

```
public class MainActivity extends AppCompatActivity {
    RecyclerView recyclerView;
    StudentAdapter adapter;
    ArrayList<Student> students;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        //thao tác với recycleView
        recyclerView = findViewById(R.id.studentsList);
        students = new ArrayList<Student>();
        students= getListStudent();
        //Khởi tạo adapter
        adapter = new StudentAdapter(students, this);
        //Hiển thị nội dung item theo chiều ngang
        LinearLayoutManager linearLayoutManager = new
LinearLayoutManager(this,
                RecyclerView.VERTICAL,false);
        //gán adapter cho recyclerView
        recyclerView.setAdapter(adapter);
        //set layout cho recyclerView
        recyclerView.setLayoutManager(linearLayoutManager);
    }

    private ArrayList<Student> getListStudent() {
```

```
        //Tự phát sinh 20 dữ liệu mẫu
        ArrayList< Student> list= new ArrayList<>();
        for (int i = 1; i <= 20; i++) {
            students.add(new Student("Student Name"+i , 2000 + (i % 2)));
        }
        return list;
    }
}
```