



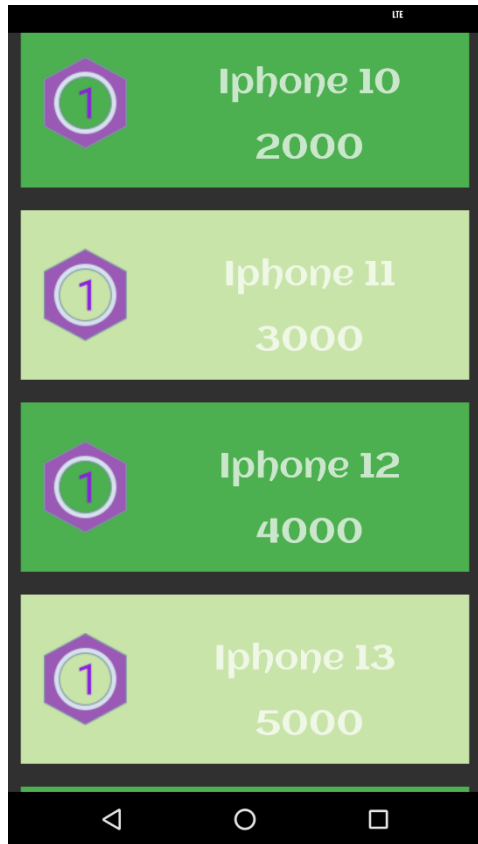
# RecyclerView

## רשימה מעוצבת

חגית כהן

[see my git links in the end. Guy](#)

# – RecyclerView הרעיון הכללי



**RecyclerView** פקד של אנדרואיד המאפשר להציג רשימה של איברים.

רשימה זו עשויה להתעדכן בזמן ריצה – הוספת /מחיקת איברים.

כמו כן, ניתן לבחור פריט מתוך רשימה וכן הלאה .  
כל איבר ברשימה יכול להכיל מספר ערכים

– ברשימת שיאים :שם שחקן, תמונה, ניקוד...

נוכל להגדיר לכל איבר הן פעולה אחרי לחיצה רגילה, והן עבור לחיצה ארוכה.

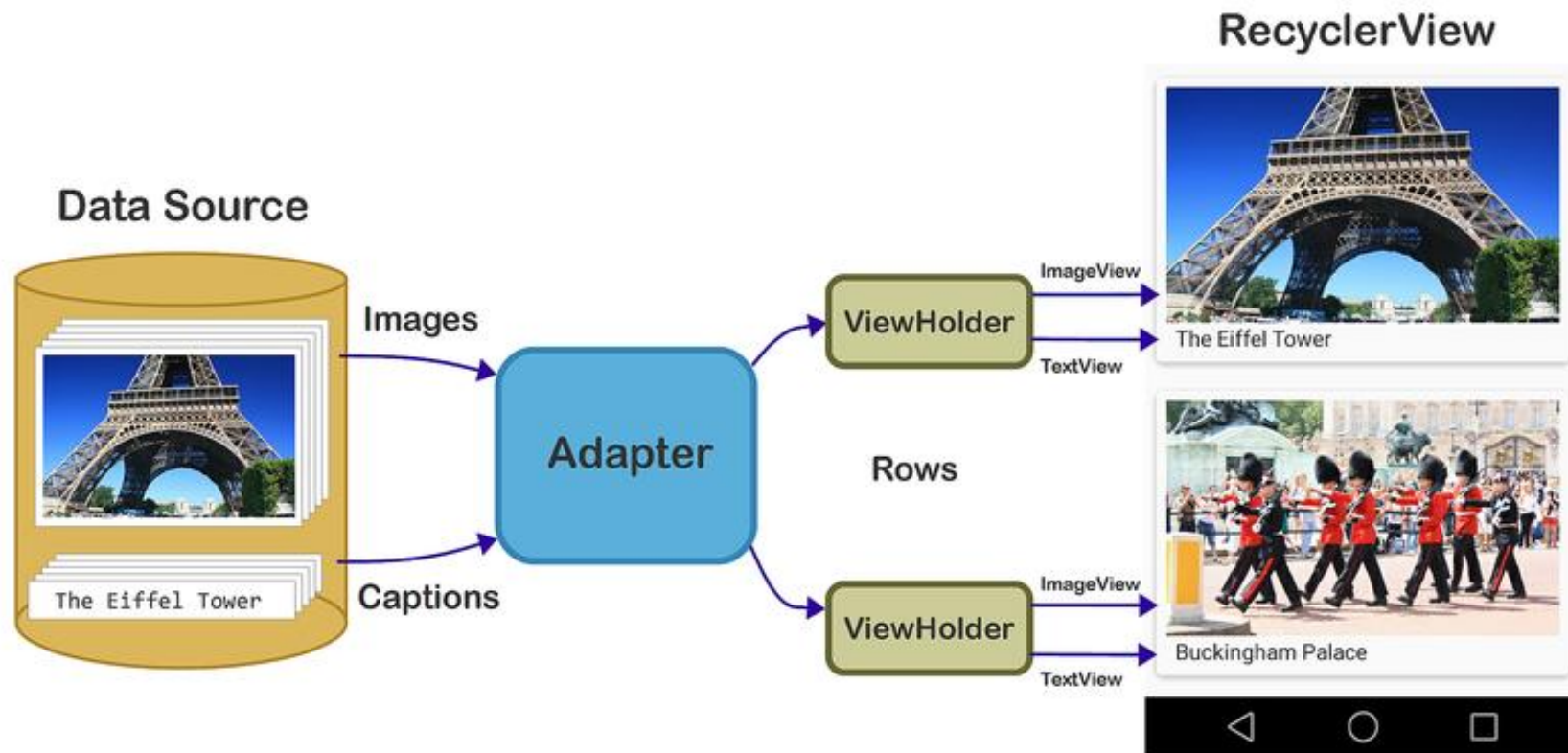
# רעיון כללי



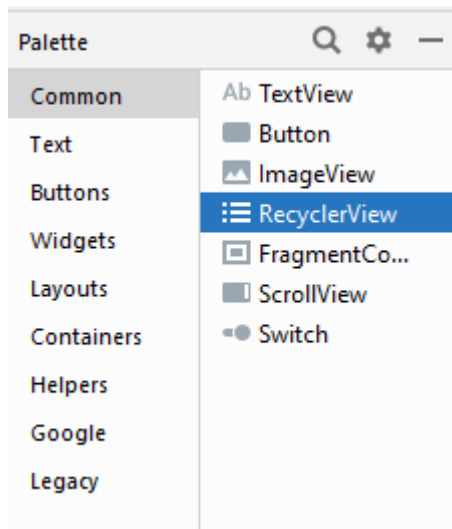
רשימה

מתאם

תצוגת רשימה



# שלב - 1 עיצוב ה Layout



1. פרווייקט חדש או מסך חדש בפרוייקט קיים.
2. נבחר RecyclerView
3. כמובן שיש לתת לכל אחד מהאלמנטים מזהה ID -

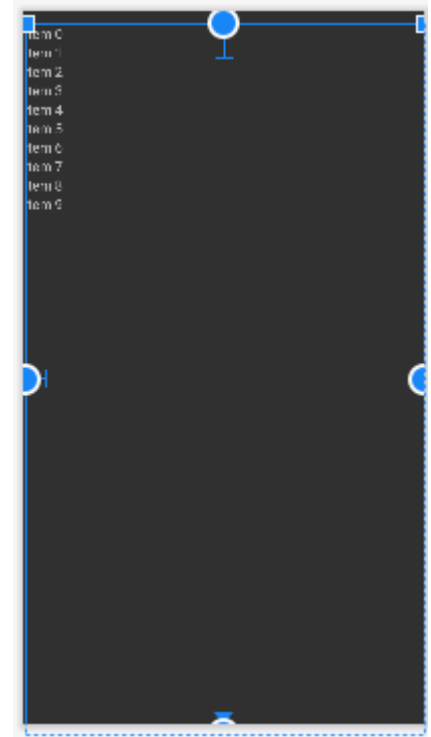
נגדיר רכיב תצוגה עליו נציג את  
הרשימה המעוצבת

# - דוגמא לעיצוב הLayout



```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    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:id="@+id/constraintLayout01"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".BoardActivity">

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



# שלב – 2קישור לפרטי תצוגה



בכדי לעבוד עם אלמנט בתצוגה (רשימה ותמונה) נקשר אותם לעצבים בקוד.

```
public class BoardActivity extends AppCompatActivity{
```

```
private RecyclerView recyclerView;
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_board);  
    recyclerView = findViewById(R.id.listView);  
}
```

ניצור אובייקט ונקשר אותו לרכיב  
התצוגה עליו נציג את הרשימה  
המעוצבת

# שלב – 3 מחלקת Item



```
package com.hagitc.make7game;

public class Item {
    private String name;
    private int price;
    private String seller;

    public Item(String name, int price, String seller) {
        this.name = name;
        this.price = price;
        this.seller = seller;
    }

    public String getName() {
        return name;
    }

    public int getPrice() {
        return price;
    }

    public String getSeller() {
        return seller;
    }
}
```

נגדיר מחלקה אותה נציג  
ברשימה. במקרה הזה בחרתי  
מחלקה עם 3 תכונות

# שלב - 4 הגדרת שורת תצוגה



```
android:id="@+id/imgRow"
```



TextView  
TextView

```
android:id="@+id/tv_name"
```

```
android:id="@+id/tv_price"
```

לכל פריט אותו נרצה  
לשנות, ניתן מזהה.

נגדיר כיצד תראה שורה  
בתצוגה – פריט  
מהרשימה



# היכרות עם ArrayList



נשתמש במבנה נתונים חדש – ArrayList.

מבנה זה מאפשר לנו ליצור מערך רשימה דינמי – אין צורך להגדיר את מספר הפריטים (רשימה (וניתן לגשת לכל פריט ברשימה) (מערך). למעשה, קיים בשפה מבנה נתונים המאחד את יתרונות המערך והרשימה

```
private ArrayList<String> myList;
```

הצהרה על רשימה. סוג הפריטים מוגדר בסוגריים משולשים

```
myList = new ArrayList<>();
```

איתחול הרשימה. אין צורך לרשום את סוג הנתונים – כבר הגדרנו אותו

```
myList.add("Banana");
```

הוספה – אחד אחרי השני לפי סדר ההכנסה

```
myList.remove(position);
```

מחיקה – לפי מיקום סידורי

```
myList.get(position);
```

החזרת פריט לפי מיקום ברשימה

# שלב – 5 יצירת רשימת פריטים



נכין רשימה של שמות אותם נראה להציג – כל תלמיד ייבחר רשימה משלו.  
בדוגמה: טלפונים.

נשתמש במבנה נתונים חדש. ArrayList –

```
public class BoardActivity extends AppCompatActivity{

    private ArrayList<Item> list;
    private RecyclerView recyclerView;
    private CustomAdapter adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_board);
        recyclerView = findViewById(R.id.listView);
        list = new ArrayList<>();
        list.add(new Item("Iphone 10", 2000,"Apple"));
        list.add(new Item("Iphone 11", 3000,"Apple"));
        list.add(new Item("Iphone 12", 4000,"Apple"));
        list.add(new Item("Iphone 13 ", 5000,"Apple"));
        list.add(new Item("One note", 1900,"Google"));
        list.add(new Item("Xiommi phone", 1900,"Xiommi"));
```

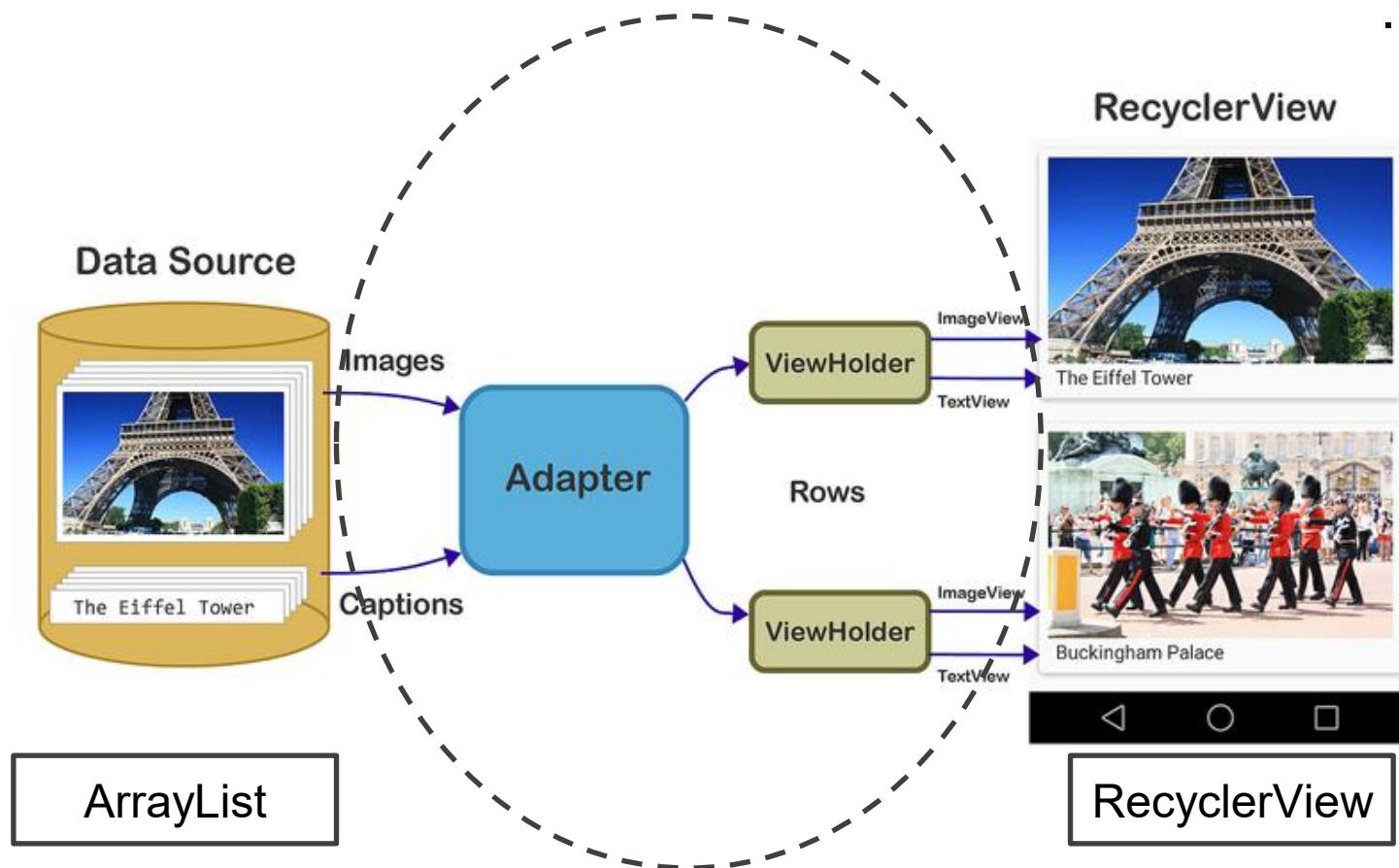
## Data Source



# 6 - Adapter שלב

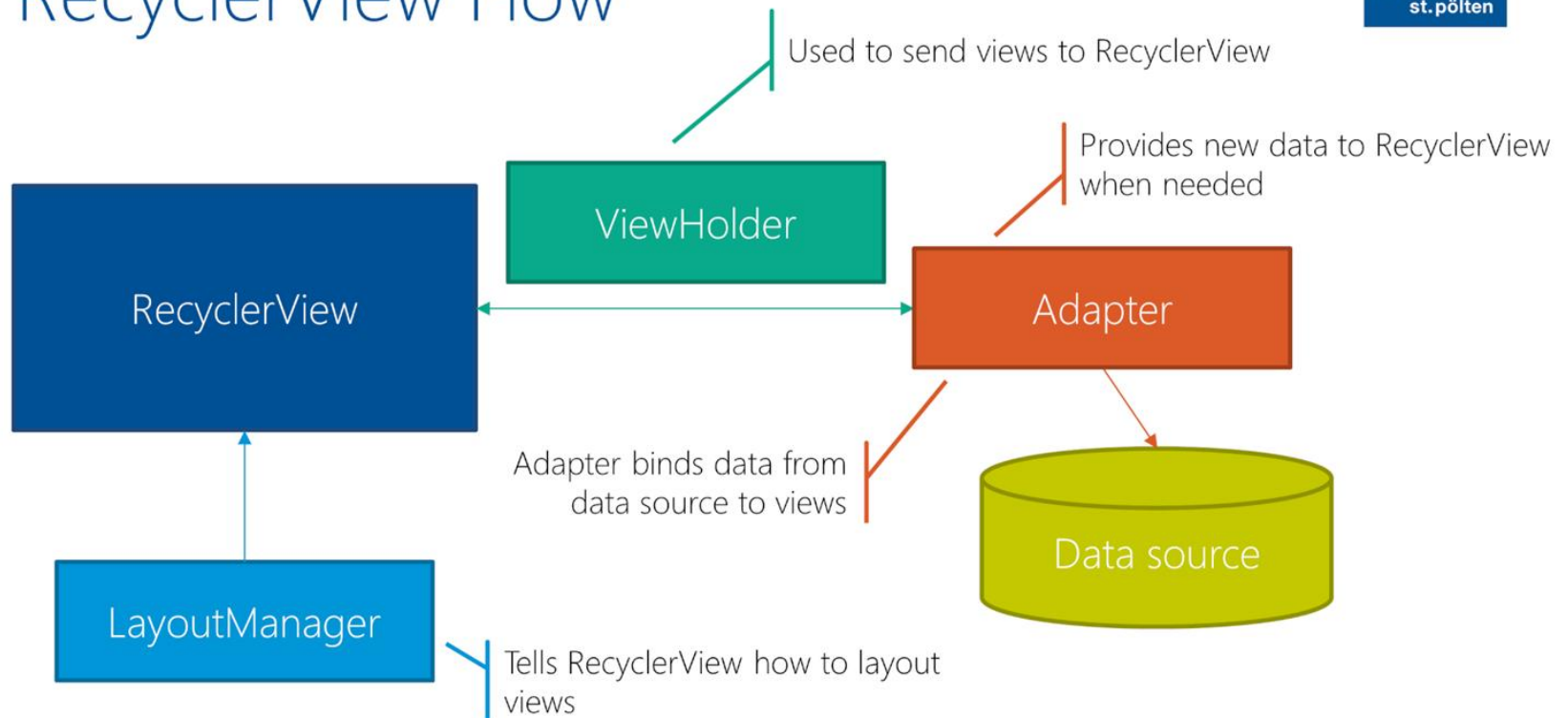


יש לנו אלמנט תצוגה RecyclerView –  
יש לנו גם רשימת פריטים ArrayList – אותה נרצה להציג  
כעת עלינו לחבר ביניהם. נשתמש במתאם אשר יגדיר כיצד נציג כל פריט  
מהרשימה.





## RecyclerView Flow



# ארכיטקטורה



```
public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.ViewHolder>
{
```

מגדיר את כל השורות ברשימה

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

מחלקה המייצגת שורה ברשימה

```
}
```



TextView

TextView

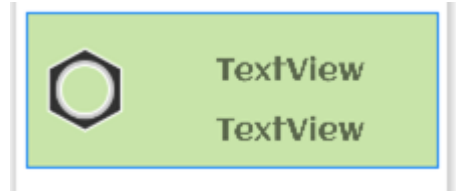
```
}
```

# מחלקת ViewHolder

## מחלקה פנימית



```
public class ViewHolder extends RecyclerView.ViewHolder {  
    public final TextView tv_name;  
    public final TextView tv_price;  
    public final ImageView img;  
    public final ConstraintLayout mainRow;
```



```
    public ViewHolder(View view) {  
        super(view);
```

הבנאי מקבל שורת תצוגה - view

```
        tv_name = view.findViewById(R.id.tv_name);  
        tv_price = view.findViewById(R.id.tv_price);  
        img = view.findViewById(R.id.imgRow);  
        mainRow = view.findViewById(R.id.mainRow);
```

```
    }  
}
```

מקשרים את העצמים שלנו לשורת

התצוגה .

נשתמש בעמצים אלו בכדי להציג נתונים

# מחלקת Adapter



```
public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.ViewHolder> {
```

```
    private ArrayList<Item> localDataSet;
```

מבנה הנתונים אותו נציג -  
נקבל אותו בבנאי

```
    public CustomAdapter(ArrayList<Item> list) {  
        localDataSet = list;  
    }
```

נשמור את הנתונים שקיבלנו

# מחלקת Adapter



```
public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.ViewHolder> {
```

```
@Override
```

```
public ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
```

```
    return viewHolder;
```

פעולה הפותחת את קובץ עיצוב השורה  
ומייצרת ממנו viewHolder

```
@Override
```

```
public void onBindViewHolder(ViewHolder viewHolder, int position)
```

```
{
```

```
}
```

```
@Override
```

```
public int getItemCount() {
```

```
    return localDataSet.size();
```

```
}
```

```
}
```

פעולה המקבלת שורת עיצוב ומיקום ברשימה  
ופורסת אותו בתצוגה

מחזיר כמה פריטים קיימים



# מחלקת Adapter



```
public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.ViewHolder> {
```

```
@Override  
public ViewHolder onCreateViewHolder
```

פעולה הפותחת את קובץ עיצוב השורה  
ומייצרת ממנו ViewHolder

```
@Override  
public void onBindViewHolder(ViewHolder viewHolder, int position)
```

פעולה המקבלת שורת עיצוב ומיקום ברשימה  
ופורסת אותו בתצוגה

```
@Override  
public int getItemCount()
```

מחזיר כמה פריטים קיימים

נייצר שורה  
ריקה

נפרוש נתונים לפי  
מיקום

נבצע מספר פעמים :  
כמספר הפריטים שיש  
ברשימה

# מחלקת Adapter

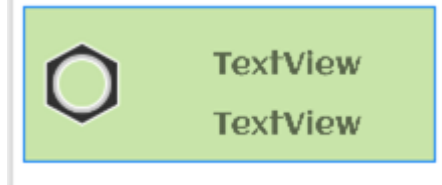


```
public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.ViewHolder> {
```

```
    private ArrayList<Item> localDataSet;
```

```
@Override
public ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
    Context context = viewGroup.getContext();
    LayoutInflater inflater = LayoutInflater.from(context);
    View contactView = inflater.inflate(R.layout.item_row, viewGroup, false);

    ViewHolder viewHolder = new ViewHolder(contactView);
    return viewHolder;
}
}
```



פותח את קובץ העיצוב  
של שורה

מייצר ממנו  
viewHolder

# מחלקת Adapter

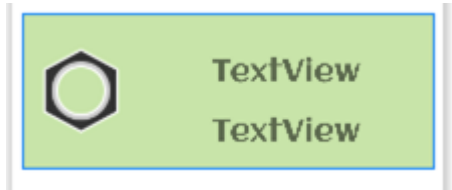


```
@Override
public void onBindViewHolder(ViewHolder viewHolder, int position)
{
    viewHolder.tv_name.setText(localDataSet.get(position).getName());
    viewHolder.tv_price.setText(""+localDataSet.get(position).getPrice());
    viewHolder.img.setImageResource(R.drawable.n1);
    if(position%2==0)
        viewHolder.mainRow.setBackgroundColor(Color.parseColor("#4CAF50"));
    else
        viewHolder.mainRow.setBackgroundColor(Color.parseColor("#FFC8E4A9"));

    viewHolder.getAdapterPosition();
}
```

מקבל viewHolder  
ומיקום ברשימה

מציג נתונים על  
התצוגה



# מחלקת Adapter



```
public class CustomAdapter extends RecyclerView.Adapter<CustomAdapter.ViewHolder> {
```

```
    @Override
```

```
    public ViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType) {
        Context context = viewGroup.getContext();
        LayoutInflater inflater = LayoutInflater.from(context);
        View contactView = inflater.inflate(R.layout.item_row, viewGroup, false);
```

```
        ViewHolder viewHolder = new ViewHolder(contactView);
        return viewHolder;
    }
```

```
    @Override
```

```
    public void onBindViewHolder(ViewHolder viewHolder, int position)
    {
```

```
        viewHolder.tv_name.setText(localDataSet.get(position).getName());
        viewHolder.tv_price.setText(""+localDataSet.get(position).getPrice());
        viewHolder.img.setImageResource(R.drawable.n1);
        if(position%2==0)
            viewHolder.mainRow.setBackgroundColor(Color.parseColor("#4CAF50"));
        else
            viewHolder.mainRow.setBackgroundColor(Color.parseColor("#FFC8E4A9"));
```

```
        viewHolder.getAdapterPosition();
    }
```

```
    @Override
```

```
    public int getItemCount() {
        return localDataSet.size();
    }
}
```

מייצר שורת תצוגה  
ריקה

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

בשקף הבא

```
}
```

עבור כל שורה, מציג  
את הנתונים מהרשימה

מחזיר כמה פריטים  
קיימים

# שלב 6 – חיבור לתצוגה



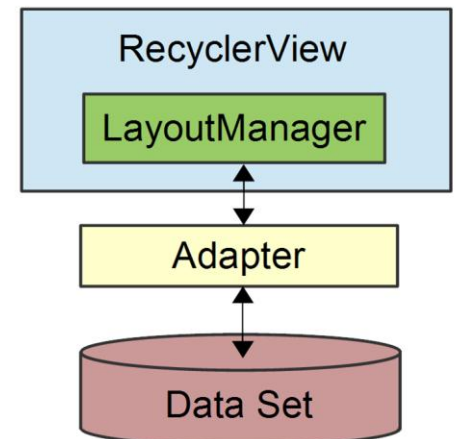
```
public class BoardActivity extends AppCompatActivity{

    private ArrayList<Item> list;
    private RecyclerView recyclerView;
    private CustomAdapter adapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_board);
        recyclerView = findViewById(R.id.listView);
        list = new ArrayList<>();
        list.add(new Item("Iphone 10", 2000, "Apple"));
        list.add(new Item("Iphone 11", 3000, "Apple"));
        list.add(new Item("Iphone 12", 4000, "Apple"));
        list.add(new Item("Iphone 13 ", 5000, "Apple"));
        list.add(new Item("One note", 1900, "Google"));
        list.add(new Item("Xiommi phone", 1900, "Xiommi"));

        adapter = new CustomAdapter(list);

        recyclerView.setAdapter(adapter);
        recyclerView.setLayoutManager(new LinearLayoutManager(this));
    }
}
```

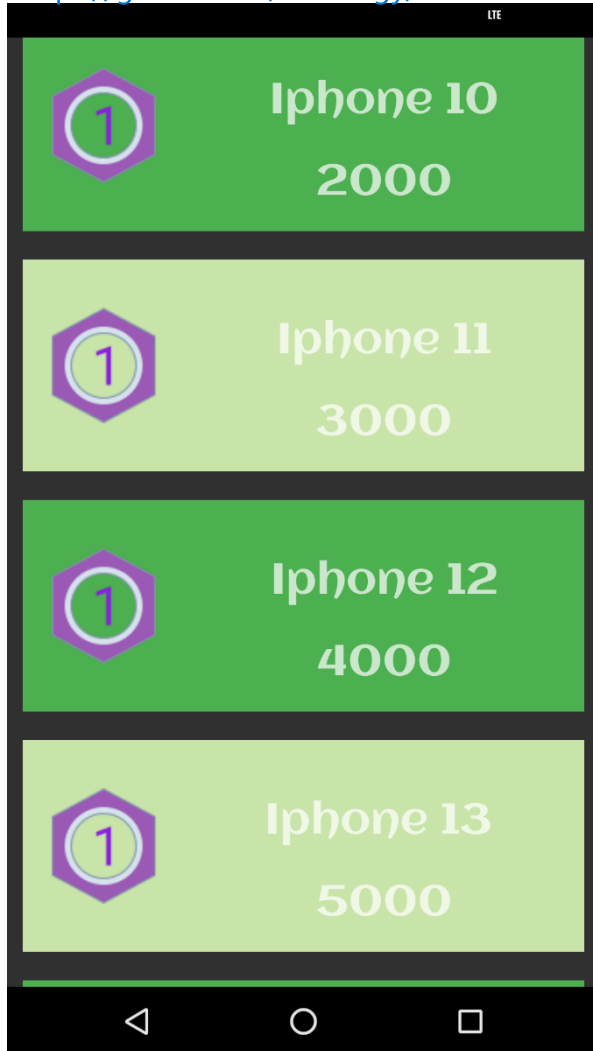


# תוצאת ריצה

<https://github.com/3strategy/recycler> = 1. complete code for this example



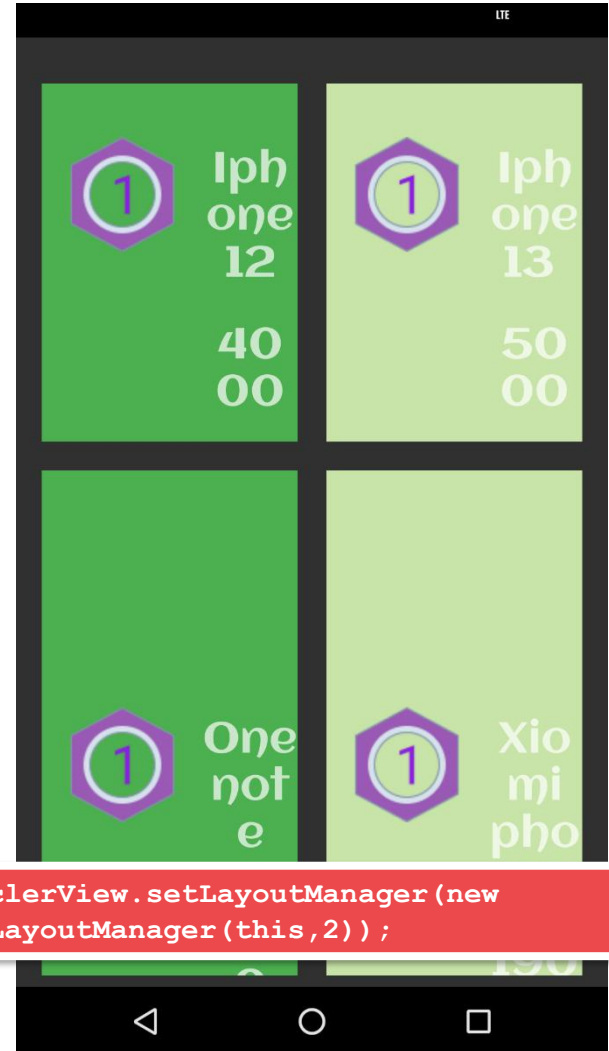
<https://github.com/3strategy/TestKtGr>



= 2. (a project with kotlin gradle, and Java main activity. Gradle is updated to allow for Kotlin activity to use JetPack compose)

3. the char

<https://claude.ai/share/c85d8da6-9939-4ce7-ad12-e320f88e8ecc>



```
recyclerView.setLayoutManager(new  
GridLayoutManager(this,2));
```