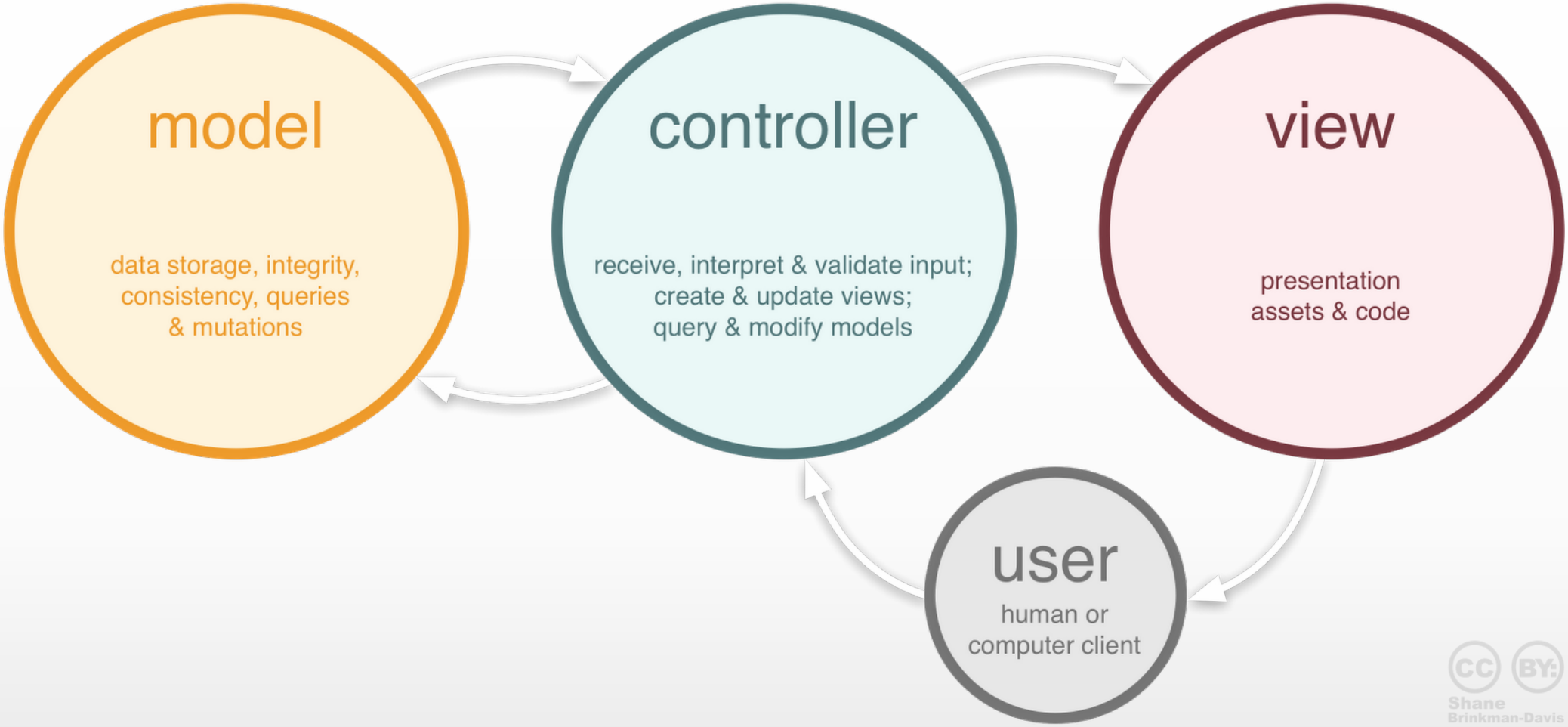
על מה נדבר

- Finding Views By Id
- Setting View properties from Code
- Practice

MVC



ניתן לחשוב על קובץ העיצוב (xml) כעל View
על ה-Activity (קובץ הMainActivity java) כעל ה-Controller
ועל המודל כעל מחלקות שנכתבות כך שאינן תלויות בממשק המשתמש כלל ומייצגות את המודל ואת הבינה העסקית של האפליקציה.

ההפרדה הזו מאפשרת כתיבת מודל שניתן לעשות בו שימוש חוזר.
ומאפשרת עבודה בנפרד (אנשים/צוותים שונים) על קבצים שונים באפליקציה. וכן סידור הגיוני לפי תפקידים.

MainActivity – The Controller

```java
package android.itomerbu.calculator;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;

public class HelloWorldActivity extends AppCompatActivity {

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

}
```

הכרות ראשונית:

אנו רואים שהמחלקה MainActivity יורשת מ-AppCompatActivity. כמו כן רואים שההתבנית המוכנה שאיתה יצרנו את האפליקציה "דורסת" עבורנו מתודה.

המתודה *onCreate* היא נקודת הכניסה שלנו לתוכנית. בהמשך נלמד על מחזור החיים של האפליקציה ועל מתודות נוספות, אך נכון לעכשיו זוהי נקודת הכניסה שלנו לתוכנית – בדומה למתודה main שהכרנו ב-Java.

השורה הראשונה במתודה onCreate קוראת למחלקה ממנה אנו יורשים ומעבירה לה את הפרמטר savedInstanceState. על כך נרחיב כשנלמד על מחזור החיים של האפליקציה.

השורה השנייה בוחרת את קובץ העיצוב ע"י פנייה למתודה setContentView

setContentView(R.layout.*activity_main*);

MainActivity – The Controller

הכרות ראשונית:

השורה השנייה במתודה *onCreate*

בוחרת את קובץ העיצוב ע"י פנייה למתודה

setContentView(R.layout.*activity_main*);

כאן אנו פוגשים לראשונה מחלקה בשם R

מחלקה זו מאפשרת גישה למשאבים

ה-resources של האפליקצייה.

עוד על המחלקה R בעמוד הבא.

המחלקה R

נזכור שבמבנה הפרוייקט הכרנו את תיקיית המשאבים - התיקייה res ולמדנו שניתן להוסיף לתוכה משאבים כדוגמת תמונות.

סביבת העבודה בונה עבורנו מחלקה בשם R ומאכלסת את המחלקה R במזהים עבור המשאבים שלנו מתיקיית res.

בתוך ה-APK של התוכנית המשאבים לא הופכים לבינארי אלא נארזים ומקבלים ID.

מזהה לכל קבצי ה-Layout שלנו יישמרו ב-R תחת ***R.layout***

מזהה לכל קבצי התמונות שלנו יישמר תחת ***R.drawable***

כך גם מזהים ל-*Strings* צבעים ועוד שנגדיר בתיקיית המשאבים יישמרו עבורנו בצורה אוטומטית במחלקה *R*.

נוצרת באופן אוטומטי ואנו *R* לא משנים את התוכן שלה.

```
package college.ness.tomer.mycirclebuttonapp;

public final class R {
    public static final class anim {
        public static final int abc_fade_in=0x7f050000;
        public static final int abc_fade_out=0x7f050001;
        public static final int abc_slide_in_bottom=0x7f050002;
        public static final int abc_slide_in_top=0x7f050003;
        public static final int abc_slide_out_bottom=0x7f050004;
        public static final int abc_slide_out_top=0x7f050005;
    }
    public static final class attr {
        /** <p>Must be a reference to another resource, in the f
or to a theme attribute in the form "<code>?[<i>package</i>:][<i
         */
        public static final int actionBarDivider=0x7f01005c;
```

```
    public static final class id {
        public static final int editText=0x7f090044;
        public static final int editText2=0x7f090042;
        public static final int editText3=0x7f090040;
        public static final int editText4=0x7f090045;
        public static final int action_bar=0x7f090031;
        public static final int action_bar_activity_content=0x7f090000;
```

```
    public static final class layout {
        public static final int activity_main=0x7f040017;
        public static final int abc_action_bar_title_item=0x7f040000;
        public static final int abc_action_bar_up_container=0x7f040001;
        public static final int abc_action_bar_view_list_nav_layout=0x7f040002;
        public static final int abc_action_menu_item_layout=0x7f040003;
        public static final int abc_action_menu_layout=0x7f040004;
    }
    public static final class menu {
        public static final int menu_main=0x7f0d0000;
    }
    public static final class mipmap {
        public static final int ic_launcher=0x7f030000;
```

MainActivity – The Controller

קישור בין הפקדים בקובץ העיצוב (View) לבין הקוד ב-Activity ה-Controller

אין קשר ישיר בין המחלקה Activity לבין קובץ העיצוב עד לשורה
setContentView(R.layout.activity_main);

כפועל יוצא מההפרדה הזו שקיימת בין ה-View לבין ה-Controller נצטרך בתהליך קישור גם עבור הפקדים.

public View **findViewById** (int id)

Finds a view that was identified by the id attribute from the XML that was processed in `onCreate(Bundle)`.

Returns

The view if found or null otherwise.

המתודה מחזירה פקד View לפי id שניתן לו בקובץ העיצוב שהוגדר בשורה setContentView.

דוגמא לשימוש במתודה:

```
ImageView imageView = (ImageView) findViewById(R.id.myimageview);
imageView.setImageResource(R.drawable.myimage);
```

מכיוון שהמתודה מחזירה View (מחלקת האב של כל הפקדים) לעיתים קרובות נבצע casting כדי לעבוד עם הפקד כמחלקה היורשת.
ברגע שיש בידנו משתנה שמייצג את הפקד – אנו יכולים לשנות את תכונות הפקד ע"י מתודות setters ולגשת למידע ששמור בפקד ע"י מתודות getters.
בדוגמא החלפנו תמונה בפקד מסוג ImageView. דוגמאות נוספות בעמוד הבא.

MainActivity – The Controller

קישור בין הפקדים בקובץ העיצוב (View) לבין הקוד ב-Activity ה-Controller

```
public class MainActivity extends ActionBarActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
            Type varName = (Type)findViewById(R.id.xmlVarID);
        Button btnCircle = (Button)findViewById(R.id.btnCircle);
        btnCircle.setText("New Text From Code!");
        btnCircle.setBackgroundColor(0xFFDD00FF);
            Type varName = (Type)findViewById(R.id.xmlVarID);
        EditText etResult = (EditText) findViewById(R.id.etResult);
        String inputText = etResult.getText().toString();
        etResult.setHint("A New Hint From Code!");
        etResult.setTextColor(0xeeddffcc);
            Type varName = (Type)findViewById(R.id.xmlVarID);
        SeekBar sbRed = (SeekBar) findViewById(R.id.sbRed);
        sbRed.setProgress(255);
    }
```

הרשמה לאירועים - Events

אירועים הם כזכור שינויים שאנו מעוניינים לדעת עליהם, אירוע יכול להיות לדוגמא:

לחיצה על כפתור, הזזה של סליידר, שינוי טקסט בתיבת טקסט, שינוי מיקום גיאוגרפי, הורדת תוכן הסתיימה ועוד.

אירועים באנדרואיד ממומשים באמצעות תבנית העיצוב *Observer* שהכרנו במודול תכנות מונחה עצמים.

הפקדים מגדירים interface (הגדרה של מתודות שיש לממש) וכן מאפשרים רישום מאזין לאירוע ע"י מתודת setter למאזין.

ברגע שהאירוע מתרחש – הפקד מודיע למאזין ע"י הפעלת המתודה של הנרשם.

דוגמא למתודה setOnClickListener של המחלקה View – לכל פקד באנרואיד יש מתודת setter למאזין עבור לחיצה:

public void **setOnClickListener** (View.OnClickListener l)

Register a callback to be invoked when this view is clicked. If this view is not clickable, it becomes clickable.

Parameters

l The callback that will run

See Also

`setClickable(boolean)`

המתודה היא כמובן מתודת מופע ולכן עבור כל מופע של פקד (לדוגמא משתנה מסוג Button שקישרנו מקובץ העיצוב) - נוכל להרשם לאירוע הלחיצה של הכפתור.

המתודה מקבלת משתנה מסוגView.OnClickListener ולכן כדי להשתמש במתודה נצטרך לממש את האינטרפייס או להשתמש במופע של מחלקה אנונימית.

הרשמה לאירועים - Events

דוגמא להרשמת מאזין אנונימי לאירוע לחיצה על כפתור:

```
public class MainActivity extends ActionBarActivity {

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

        Button btnCircle = (Button)findViewById(R.id.btnCircle);

        View.OnClickListener btnListener = new View.OnClickListener() {
            @Override
            public void onClick(View v) {

            }
        };

        btnCircle.setOnClickListener(btnListener);
    }
```

המתודה שתגיב לאירוע

יתרונות מאזין אנונימי:
קל לכתיבה. שימושי במידה והמתודה מגיבה אך ורק לאירוע לחיצה של כפתור אחד.

חסרונות:
ברגע יצירת מופע של מחלקה פנימית אנונימית – המשתנים של המתודה מועתקים למחלקה הפנימית. לכן מותר להשתמש אך ורק בקבועים (final) מתוך המחלקה האנונימית. כמו כן this מצביע על המופע של המחלקה האנונימית. ניתן לגשת ל MainActivity.this

הרשמה לאירועים - Events

הרשמת מאזין אנונימי לאירוע לחיצה על מספר כפתו

```
public class MainActivity extends ActionBarActivity {
    private int x = 0;
    Button btn1, btn2;

    View.OnClickListener listener = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            switch(v.getId()){
                case R.id.btn1:{
                    x+=1;
                    break;
                }
                case R.id.btn2:{
                    x+=2;
                    break;
                }
            }
            System.out.println(x);
        }
    };
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        System.out.println(x++);

        initLayout();
        initEvents();
    }

    private void initLayout() {
        btn1= (Button) findViewById(R.id.btn1);
        btn2= (Button) findViewById(R.id.btn2);
    }

    private void initEvents() {
        btn1.setOnClickListener(listener);
        btn2.setOnClickListener(listener);
    }
}
```

המתודה שתגיב לאירוע

מאזין אנונימי יכול גם לשמש לאירוע של מס' פקדים.
במקרה זה נצטרך לשמור את המופע של המחלקה האנונימית כמשתנה.

לדוגמא:

```
View.OnClickListener listener = new View.OnClickListener() {
    @Override
```

יתרונות :
קל לכתיבה. שימושי במידה והמתודה מגיבה לאירוע של מס' פקדים.

חסרונות:
דורש סנכרון במידה ועובדים ב-MultiThreading
לא מסודר. דורש משפט תנאי/switch לבדיקה.

הרשמה לאירועים - Events

הרשמת המחלקה הנוכחית כמאזין לאירוע הלחיצה:

```
public class MainActivity extends ActionBarActivity implements View.OnClickListener {
    private int x = 0;
    Button btn1, btn2;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        System.out.println(x++);

        initLayout();
        initEvents();
    }

    private void initLayout() {
        btn1= (Button) findViewById(R.id.btn1);
        btn2= (Button) findViewById(R.id.btn2);
    }

    private void initEvents() {
        btn1.setOnClickListener(this);
        btn2.setOnClickListener(this);
    }

    @Override
    public void onClick(View v) {
        switch(v.getId()){
            case R.id.btn1:{
                x+=1;
                break;
            }
            case R.id.btn2:{
                x+=2;
                break;
            }
        }
        System.out.println(x);
    }
}
```

המתודה שתגיב לאירוע

המחלקה הנוכחית/ה-Controller יכולה לממש את האינטרפייס וכך להאזין לאירועים.

לדוגמא:

```
public class MainActivity extends ActionBarActivity implements View.OnClickListener {
```

```
private void initEvents() {
    btn1.setOnClickListener(this);
    btn2.setOnClickListener(this);
}
```

יתרונות :

קל לכתיבה. שימושי במידה והמתודה מגיבה לאירוע של מס' פקדים.

חסרונות:

מאפשר ליצור רק מאזין אחד מכל סוג של אינטרפייס.