Изпитна тема № 15: **Приложения с графичен потребителски интерфейс**

Среди за визуално програмиране и за разработване на приложения за мобилни устройства. Основни компоненти/контроли на графичния интерфейс. Класове, методи и свойства. Основни по-нятия – събитие, обработка, източник. Работа с обекти и свързани с тях събития. Използване на конструкции за контрол на изпълнението. Прихващане и обработка на грешки. Изграждане на потребителски интерфейс на мобилно приложение. Създаване на различни екрани и връзка между тях. Работа с изображения и текст. Достъп до хардуера на мобилно устройство. Изграждане на приложения с определен модел база данни. Сигурност на мобилното приложение. Тестване и публикуване на мобилно приложение.

Критерии за оценяване на изпитна тема № 15	**Максимален брой точки**
1. Дефинира понятията: интегрирана среда за разработка (IDE), графичен интерфейс, събитие, обработка, източник. Различава основни контроли на графичния интерфейс и посочва техните свойства.	6
2. Дава пример за конструкции за контрол на изпълнението. Демонстрира употребата на конструкции за прихващане и обработка на изключения.	10
3. Демонстрира работа с класове, обекти и свързаните с тях събития.	6
4. Обяснява моделите бази данни. Разработва модел бази данни, така че да реши поставената задача.	10
5. Посочва и различава видовете мобилни операционни системи и съответните платформи за създаване на мобилни приложения. Определя възможността за разработване на приложение според възможностите на мобилното устройство.	14
6. Познава начините за работа с текст и изображения в приложение за мобилно устройство.	2
7. Дава пример за създаване на различни екрани в мобилно приложение и връз-ката между тях.	4
8. Демонстрира знания, чрез които осъществява достъп до хардуера на мобилно устройство.	6
9. Обяснява принципите на сигурност при проектиране и разработване на мобилни приложения.	4
10. Избира подходящи методи за тестване на мобилно приложение, така че да реши поставената задача.	8
11. Открива грешки в програмен код и го модифицира, така че да реши поста-ве-ната задача.	6
12. Анализира, определя и допълва програмен код, така че да реши поставената задача.	24
ОБЩ БРОЙ ТОЧКИ:	**100**

1- **Дефинира понятията: интегрирана среда за разработка (IDE), графичен интерфейс, събитие, обработка, източник. Различава основни контроли на графичния интерфейс и посочва техните свойства.(6т.)**

IDE- софтуерно приложение помагащо на програмисти да развиват софтуера си, повишавайки ефективноста и добавяйки елементи като testing, build, edit, packaging и тн.

Графичен интерфейс- интерфейс, в който елементите, предоставени на потребителя за управление, са изпълнени във вид на графични изображения (менюта, бутони, списъци и др.).

Събитие- В програмирането има действия, наречени събития, които предизвикват други действия в програмата. Събитието може да е предизвикано от потребителя с натискане на конкретен бутон, кликване на мишката или др. Може и да е резултат от изпълнението на самата програма.

Обработка- (прихващане на събитието) За обработка на дадено събитие трябва да има свързан с него слушател. И слушателя е този който описва какво да се случи при това събитие.

Източник- Обектът, който предизвиква дадено събитие, се нарича изпращач на събития (event sender)

Основни контроли на графичния интерфейс- TextBox, Label, Button и тн.

TextBox - за извеждане на текст на интерфейса,

Button - създаване на бутон с който можем да създадем събитие(примерно при натискане)

Label - за подаване на пояснителен текст към нещо по интерфейса

2- **Дава пример за конструкции за контрол на изпълнението. Демонстрира употребата на конструкции за прихващане и обработка на изключения.(10т.)**

ВВ програмирането, контролът на производителността се постига чрез различни техники и стратегии, които имат за цел оптимизиране на производителността на програмата. Ето някои общи конструкции и техники, използвани за контрол на производителността в програмирането:

Профилиране: Профилирането е процесът на анализиране на производителността на програма, като се измерват различни метрики за производителност, като време за изпълнение, употреба на процесора, употреба на памет и други. Чрез идентифициране на факторите, които намаляват производителността, разработчиците могат да оптимизират кода, за да подобрят общата производителност.

Структури от данни: Изборът на правилните структури от данни може да има значителен принос за производителността на програмата. Например използването на хеш таблица за бързи търсения или на опашка с приоритети за ефективно сортиране може да подобри производителността на програмата.

Алгоритми: Алгоритмите са от решаващо значение за производителността на програмата. Използването на ефективен алгоритъм може значително да намали времето за изпълнение на програмата. Разработчиците трябва винаги да избират правилния алгоритъм за конкретната задача.

Паралелизъм: Паралелизмът включва изпълнението на множество задачи едновременно, за да се подобри производителността. Мултипоточността и мултипроцесорността са някои от техниките, използвани за паралелизъм. Чрез разделянето на работното натоварване между множество нишки или процеси разработчиците могат да подобрят производителността на програмата.

…

-употреба на прихващане и обработка на изключения

```
try
        {
            file.ReadBlock(buffer, index, buffer.Length);
        }
        catch (System.IO.IOException e)
        {
            Console.WriteLine("Error reading from {0}. Message = {1}",
path, e.Message);
        }
        finally
        {
            if (file != null)
            {
                file.Close();
            }
        }
```

```
class TryFinallyTest
{
    static void ProcessString(string s)
    {
        if (s == null)
        {
            throw new ArgumentNullException(paramName: nameof(s), message:
"parameter can't be null.");
        }
    }

    public static void Main()
    {
        string s = null; // For demonstration purposes.

        try
        {
            ProcessString(s);
        }
        catch (Exception e)
        {
            Console.WriteLine("{0} Exception caught.", e);
        }
    }
}
```

```
/*
Output:
System.ArgumentNullException: Value cannot be null.
   at TryFinallyTest.Main() Exception caught.
 * */
```

3- Демонстрира работа с класове, обекти и свързаните с тях събития.(6т.)

```
void Page_Load(Object sender, EventArgs e)
    {
// Manually register the event-handling method for
// the Click event of the Button control.
Button1.Click += new EventHandler(this.GreetingBtn_Click);
}

void GreetingBtn_Click(Object sender,
EventArgs e)
{
// When the button is clicked,
// change the button text, and disable it.

Button clickedButton = (Button)sender;
clickedButton.Text = "...button clicked...";
clickedButton.Enabled = false;

GreetingLabel.Visible = true;
}
```

```
private void checkBox1_Click(object sender, System.EventArgs e)
{
   // The CheckBox control's Text property is changed each time the
   // control is clicked, indicating a checked or unchecked state.
   if (checkBox1.Checked)
   {
      checkBox1.Text = "Checked";
   }
   else
   {
      checkBox1.Text = "Unchecked";
   }
        }
```

4- Обяснява моделите бази данни. Разработва модел бази данни, така че да реши поставената задача.(10т.)

Моделите на бази данни представляват начин за организиране и представяне на данни в една база данни. Ето няколко от най-известните модели на бази данни:

йерархичен модел - този модел представя данните в дървовидна йерархия, където всеки елемент може да има множество наследници, но само един родител. Този модел обикновено се използва в стари системи.

- мрежов модел - този модел е подобен на йерархичния, но позволява на един елемент да има множество родители и наследници. Този модел е по-гъвкав от йерархичния, но все още е малко ограничен.
- релационен модел - този модел е базиран на таблиците, които съдържат данни във вид на релации. Релациите могат да бъдат свързани помежду си, като по този начин се формират сложни свързани структури на данни. Този модел е много популярен в съвременните бази данни.
- обектно-ориентиран модел - този модел се базира на обектите и класовете. Данните се организират в класове, които имат свойства и методи. Този модел е по-сложен от релационния, но може да бъде по-гъвкав за работа с обекти.

Тези са само някои от моделите на бази данни, които могат да се използват за организиране на данните в една база данни. Всеки модел има своите предимства и недостатъци, и изборът на модел зависи от конкретните нужди и цели на проекта.

Задача?

Обяснява моделите бази данни. Разработва модел бази данни, така че да реши поставената задача, **за съхранение на данните в метода OnActivityResult на val3 в база, като се използва класа след метода.**

```
protected override void OnActivityResult(int requestCode, [GeneratedEnum]
Result resultCode, Intent data)
        {
            base.OnActivityResult(requestCode, resultCode, data);
            float val3 = data.GetFloatExtra("key3", 0);
            textView1.Text = val3.ToString();
            SQLiteConnection db;
            string dbPath = Path.Combine(

Environment.GetFolderPath(Environment.SpecialFolder.Personal),
                 "database.db");
            db = new SQLiteConnection(dbPath);
            db.CreateTable<Measurement>();
            {
                var measurement = new Measurement();
                measurement.Value = val3;
                db.Insert(measurement);
            }
        }

        [Table("……s")]
        public class Measurement
        {
            [……, AutoIncrement]
            public …… Id { get; set; }
            public …… …… { get; set; }
        }
```

5- **Посочва и различава видовете мобилни операционни системи и съответните платформи за създаване на мобилни приложения. Определя възможността за разработване на приложение според възможностите на мобилното устройство.(14т.)**

-мобилни ос – Android, IOS, BlackBerry, Harmony OS(Huawei OS)

- **Android** е операционна система за мобилни устройства поддържана от Google. Операционната система поддържа различни хардуерни платформи, вкл. телефони, таблети, модни аксесоари (Android Wear), Телевизори (Android TV), автомобилни инфотеймънт системи (Android Auto) и др.
- **iOS** е мобилна операционна система на компанията Apple Inc. Платформата предлага високо ниво на защита, както на личните данни, така и на самата система, включително и чрез достъп с пръстов отпечатък. Налична и версия за използване в автомобили наречена Apple CarPlay. Платформата не е с отворен код, което ограничава до известна степен разпространението и само до устройства разработка на Apple.
- **Windows Phone/Mobile** е мобилна операционна система на Microsoft. Тя също не е с отворен код, но е достъпна на хардуерни устройства от множество производители на преносими устройства, като HTC, Samsung, Toshiba, Nokia и др. Други – BlackBerry, Tizen, Ubuntu Touch

Платформи – Android Studio, Visual Studio

На базата на функциалността на приложението избираме версията на операционната система на която ще разработваме проекта. При създаване на проекта на Android Studio ни се предлага коя версия да ползваме така че функциалността и процента устройства на които може да работи приложението са оптимални.

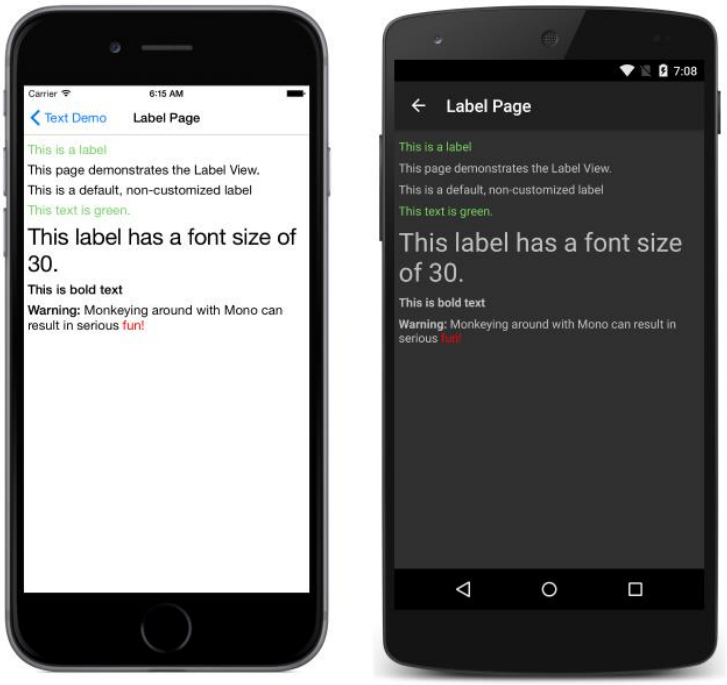
6- **Познава начините за работа с текст и изображения в приложение за мобилно устройство.(2т.)**

```
<EditText
        android:id="@+id/edittext"
        android:layout_width="match_parent"
        android:imeOptions="actionGo"
        android:inputType="text"
        android:layout_height="wrap_content" />
```

```
EditText edittext = FindViewById<EditText>(Resource.Id.edittext);
```

Ползва се TextView за показване на текст по екрана

Може да ползва и Label




```
<ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="300dp"
        android:src="@drawable/Icon" />
```

```
var imageView = FindViewById < ImageView > (Resource.Id.imageView);
```

```
<ImageSwitcher
  android:id="@+id/imageSwitcher1"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_centerHorizontal="true"
  android:layout_centerVertical="true" >
</ImageSwitcher>
```

7- Дава пример за създаване на различни екрани в мобилно приложение и връзката между тях.(4т.)

Създаваме нов Layout. Можем да сменяме между екраните при някакво събитие чрез `setContentView(R.layout.layout2)`

Примерно при натискане на бутон

Можем да ползваме ViewFlipper

```
ViewFlipper viewFlipper = (ViewFlipper) findViewById(R.id.myViewFlipper);
```

```
// you can switch between next and previous layout and display it
viewFlipper.showNext();
viewFlipper.showPrevious();

// or you can switch selecting the layout that you want to display
viewFlipper.setDisplayedChild(1);
viewFlipper.setDisplayedChild(viewFlipper.indexOfChild(findViewById(R.id.secondLayout)
```

```
<ViewFlipper
        android:id="@+id/myViewFlipper"
        android:layout_width="match_parent"
        android:layout_height="match_parent" >

        <LinearLayout
            android:id="@+id/firstLayout"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical" >
           [...]
        </LinearLayout>

        <LinearLayout
            android:id="@+id/secondLayout"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical" >
           [...]
        </LinearLayout>

        <LinearLayout
            android:id="@+id/thirdLayout"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical" >
          [...]
        </LinearLayout>
 </ViewFlipper>
```

8- Демонстрира знания, чрез които осъществява достъп до хардуера на мобилно

устройство.(6т.)

Мобилните устройства имат различни сензори и нашите програми могат да се възползват от тях. За тази цел първо ни трябва клас SensorManager и клас Sensor, както и да реализираме интерфейс SensorEventListener. Мениджъра на сензори първо изисква системна услуга за използване на сензорите. След това обект от класа Sensor си присвоява вид сензор чрез мениджъра на сензори. После методите onResume и onPause съответно регистрират и отписват сензори чрез мениджъра на сензорите. Накрая метода onSensorChanged следи за стойността на сензора и извършва действие според нея.

Ето го кода за дизайна на програмата:

```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/black"
        android:textAlignment="center"
        android:text="@string/info"
        android:textSize="40sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.32"
        android:gravity="center_horizontal" />

    <TextView
        android:id="@+id/showValue"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:text="@string/tv_text"
        android:textAlignment="center"
        android:textColor="@color/purple_500"
```

```
        android:textSize="30sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView1"
        app:layout_constraintVertical_bias="0.04" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Ето го и кода за логиката:

```
import androidx.appcompat.app.AppCompatActivity;

import android.annotation.SuppressLint;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.Bundle;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity implements
SensorEventListener {

    SensorManager sensorManager;
    Sensor lightSensor;
    TextView textView;

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

        sensorManager =
(SensorManager)getSystemService(SENSOR_SERVICE);
        lightSensor =
sensorManager.getDefaultSensor(Sensor.TYPE_LIGHT);

        textView = findViewById(R.id.showValue);
    }

    @SuppressLint("SetTextI18n")
    @Override
```

```
    public void onSensorChanged(SensorEvent sensorEvent) {
        if(sensorEvent.sensor.getType() == Sensor.TYPE_LIGHT){
            textView.setText(sensorEvent.values[0] + " lx");
        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int i) {

    }

    @Override
    protected void onResume() {
        super.onResume();
        sensorManager.registerListener(this, lightSensor,
SensorManager.SENSOR_DELAY_NORMAL);
    }

    @Override
    protected void onPause() {
        super.onPause();
        sensorManager.unregisterListener(this);
    }
}
```

9. Обяснява принципите на сигурност при проектиране и разработване на мобилни приложения. (4т.)

За да може нашето приложение да е сигурно, трябва да съобразим някои принципи за безопасност. Трябва комуникацията да е защитена, да питаме кое приложение да се използва при възможност две да извършат една и съща работа, да не позволяваме други приложения нежелано да достъпват нашето, да питаме потребителя за автентикация преди показване на чувствителна информация, да използваме SSL при уеб комуникация, да изискваме точните разрешения, ако друго приложение има разрешение да извърши дадено действие за нас, приложението ни да го използва, вместо нашето приложение да изисква същото разрешение. Трябва да съхраняваме данните безопасно, да пазим данните локално, да проверяваме за наличие на памет, да проверяваме валидността на данните, да съхраняваме само нечувствителна информация в кеша и да обновяваме услугите.

10. Избира подходящи методи за тестване на мобилно приложение, така че да реши поставената задача.

за съхраняване на данните и тяхното четене на ниво модел.

```
namespace UITest1
{
    public class Tests
    {
        [Test]
        public void ABCTest()
        {



            Assert.AssertEquals(x, y);
        }
    }
}
```

11. Открива грешки в програмен код и го модифицира, така че да реши поставената задача.

за използване на сензори.

```
        SensorManager sensorManager;
        Sensor sensor;
        float max;
        sensor = sensorManager.GetDefaultSensor(SensorType.Light);
        sensorManager.RegisterListener(this, sensor,
   SensorDelay.Fastest);
        max = sensor.MaximumRange;
        sensorManager =
   (SensorManager)GetSystemService(Context.SensorService);
```

12. **Анализира, определя и допълва програмен код, така че да реши** поставената задача.

Коментира програмения код в метода button1.Click и button2.Click. Да модефицира в метода button2.Click, така че да реши поставената задача за записване на числото f, което се получава след умножение на числата a и b, като key3.

```
[Activity(Label = "Activity1", MainLauncher = true)]
    public class Activity1 : Activity
    {
        EditText editText1;
        EditText editText2;
        TextView textView1;
        int rc = 101;
        protected override void OnCreate(Bundle savedInstanceState)
        {
            base.OnCreate(savedInstanceState);
            SetContentView(Resource.Layout.layout1);
            editText1 = FindViewById<EditText>(Resource.Id.editText1);
            editText2 = FindViewById<EditText>(Resource.Id.editText2);
            textView1 = FindViewById<TextView>(Resource.Id.textView1);
            Button button1 = FindViewById<Button>(Resource.Id.button1);
```

```
        button1.Click += (sender, e) =>
        {
            int a = int.Parse(editText1.Text);
            int b = int.Parse(editText2.Text);
            Intent intent = new Intent(this, typeof(Activity1));
            intent.PutExtra("key1", a);
            intent.PutExtra("key2", b);
            StartActivityForResult(intent, rc);
        };
        Button button2 = FindViewById<Button>(Resource.Id.button2);
        button2.Click += (sender, e) =>
        {
            Intent intent = Intent;
            int val1 = intent.GetIntExtra("key1", 0);
            int val2 = intent.GetIntExtra("key2", 0);
            float c = val1 . val2;
            Intent resultIntent = new Intent();
            resultIntent.PutExtra("key3", c);
            SetResult(Result.Ok, resultIntent);
            Finish();
        };
    }
      }
```