面向对象的软件构造实践

实验二

2022春

哈尔滨工业大学（深圳）

本学期实验总体安排

实验项目	一	二	三	四	五	六	七	八
学时数	2	2	2	4	2	2	2	4
实验内容	功能分析	单机开发		中期检查	网络功能开发		系统调试及优化	结题验收
提交内容	开题报告							PPT及结题报告

代码移植

1、获取屏幕宽高，绘制滚动的背景图片；

GameView

2

```
public void draw() {
    canvas = mSurfaceHolder.lockCanvas();
    if(mSurfaceHolder == null || canvas == null){
        return;
    }

    //绘制背景，图片滚动
    canvas.drawBitmap(ImageManager.BACKGROUND1_bg, left: 0, top: this.backGroundTop-screenHeight,mPaint);
    canvas.drawBitmap(ImageManager.BACKGROUND1_bg, left: 0,this.backGroundTop,mPaint);
    backGroundTop +=1;
    if (backGroundTop == screenHeight)
        this.backGroundTop = 0;
```

1 MainActivity→onCreate

```
public void getScreenHW(){
    //定义DisplayMetrics 对象
    DisplayMetrics dm = new DisplayMetrics();
    //取得窗口属性
    getWindowManager().getDefaultDisplay().getMetrics(dm);

    //窗口的宽度
    screenWidth= dm.widthPixels;
    Log.i(TAG, msg: "screenWidth : " + screenWidth);

    //窗口高度
    screenHeight = dm.heightPixels;
    Log.i(TAG, msg: "screenHeight : " + screenHeight);
}
```

代码移植

2.导论中图片管理类BufferedImage，在Android中要替换为Bitmap

```
public class ImageManager {

    /** 类名-图片 映射，存储各基类的图片 <br> ...*/
    private static final Map<String, BufferedImage> CLASSNAME_IMAGE_MAP = new HashMap<>();

    public static BufferedImage BACKGROUND_IMAGE;
    public static BufferedImage HERO_IMAGE;
```

```
static {
    try {

        BACKGROUND_IMAGE = ImageIO.read(new FileInputStream( name: "src/images/bg.jpg"));
        HERO_IMAGE = ImageIO.read(new FileInputStream( name: "src/images/hero.png"));
        MOB_ENEMY_IMAGE = ImageIO.read(new FileInputStream( name: "src/images/mob.png"));
        BOSS_ENEMY_IMAGE = ImageIO.read(new FileInputStream( name: "src/images/boss.png"));
```

代码移植

Android中ImageManager.java类中使用Bitmap声明对象

```
public static Bitmap BACKGROUND1_IMAGE;
public static Bitmap BACKGROUND2_IMAGE;
```

Android中GameView.java类的构造函数中调用loading_img()加载图片

```
//加载图片的方法
public void loading_img(){
    ImageManager.BACKGROUND1_bg = BitmapFactory.decodeResource(getResources(),R.drawable.bg);
    ImageManager.HERO_IMAGE = BitmapFactory.decodeResource(getResources(), R.drawable.hero);
```

代码移植

3、多线程并发时这一段代码可能存在问题：

```
private void paintImageWithPositionRevised(Graphics g, List<? extends AbstractFlyingObject> objects) {
    if (objects.size() == 0) {
        return;
    }
    for (AbstractFlyingObject object : objects) {
        BufferedImage image = object.getImage();
        assert image != null : objects.getClass().getName() + " has no image! ";
        g.drawImage(image,  x: object.getLocationX() - image.getWidth() / 2,
                y: object.getLocationY() - image.getHeight() / 2,  observer: null);
    }
}
```

程序崩溃报错信息：

```
2022-05-11 21:08:56.810 22884-22915/com.example.myplanedemo E/eglCodecCommon: GoldfishAddressSpaceHostMemor
2022-05-11 21:09:09.083 22884-22917/com.example.myplanedemo E/AndroidRuntime: FATAL EXCEPTION: Thread-2
    Process: com.example.myplanedemo, PID: 22884
    java.util.ConcurrentModificationException
        at java.util.LinkedList$ListItr.checkForComodification(LinkedList.java:966)
        at java.util.LinkedList$ListItr.next(LinkedList.java:888)
        at com.example.application.game.BaseGame.paintImageWithPositionRevised(BaseGame.java:611)
        at com.example.application.game.BaseGame.draw(BaseGame.java:214)
        at com.example.application.game.BaseGame.run(BaseGame.java:590) <1 internal call>
```

代码移植

ConcurrentModificationException：
意思就是并发修改异常，存在于并发使用 Iterator 时出现的时候。
修改方法：不使用用Iterator方法

```
private void paintIamgeWithPositionRevised(Canvas canvas, List<? extends AbstractFlyingObject> objects){
    if(objects != null && objects.size() == 0){
        return;
    }else{
        for (int i=0;i<objects.size();i++){
            Bitmap image = objects.get(i).getImage();

            assert image != null : objects.getClass().getName() + " has no image";
            canvas.drawBitmap(image,  left: objects.get(i).getLocationX() - image.getWidth()/2,  top: objects.get
        }
    }
}
```

补充

- Android Studio断点调试
- 基本调试流程
- Android Studio日志工具
- 空指针

断点调试（1）

1、安装模拟器

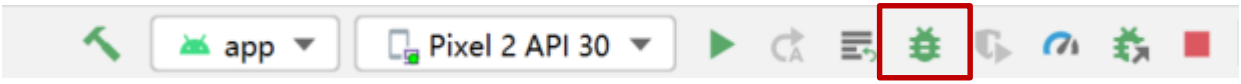
2、直接开启调试模式

① 设置断点

② Debug模式启动APP（Run → Debug）

或者点击下图红色方框得图标，Debug模式启动APP。

③ APP启动后，运行至第一处断点处会停下来，同时IDE下方出现Debug视图；同时也能看到，设置断点的代码行处的变量在监控之中。

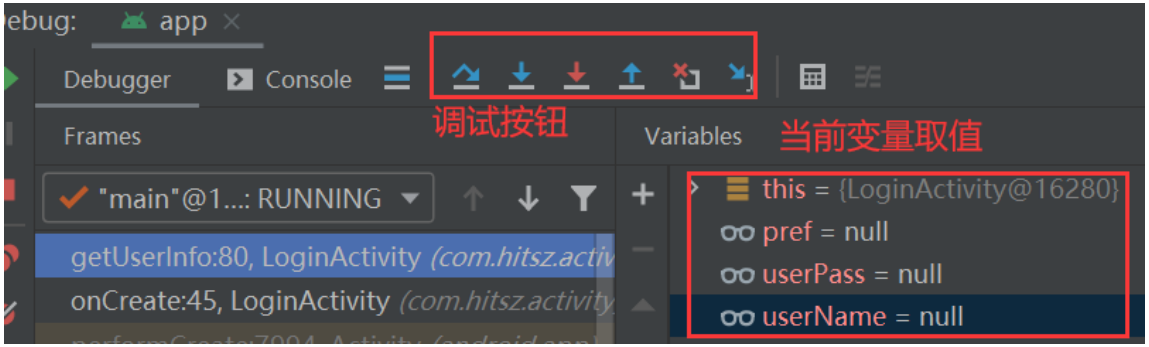



```
private SharedPreferences pref;   pref: null
private void getUserInfo(){
    pref = getSharedPreferences( name: "userInfo", MODE_PRIVATE);   pref: null
    userName = pref.getString( key: "userName",  defValue: "");
    userPass = pref.getString( key: "userPass",  defValue: "");
}
```

断点调试（1）

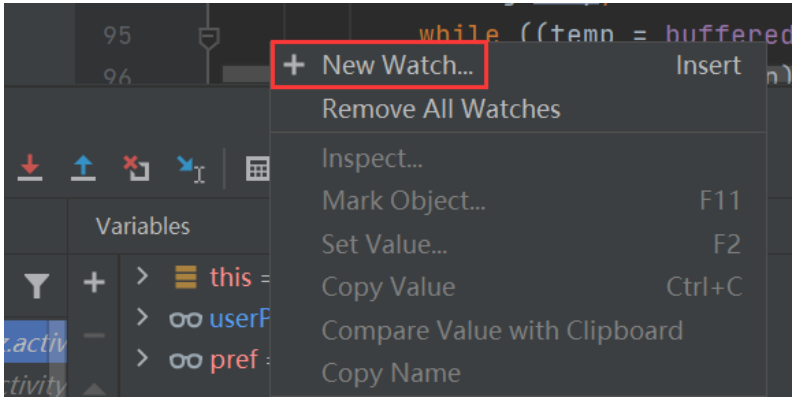
3、单步调试 step over（F8）

红色方框中的step over按钮， 程序向下执行一行；



4、Watches

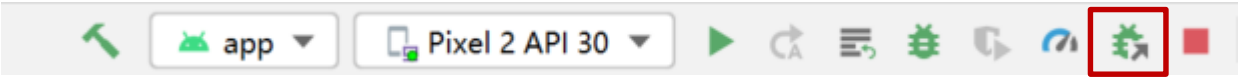
点击New Watche旁边的＋号，手动输入变量即可添加变量查看变量的值。

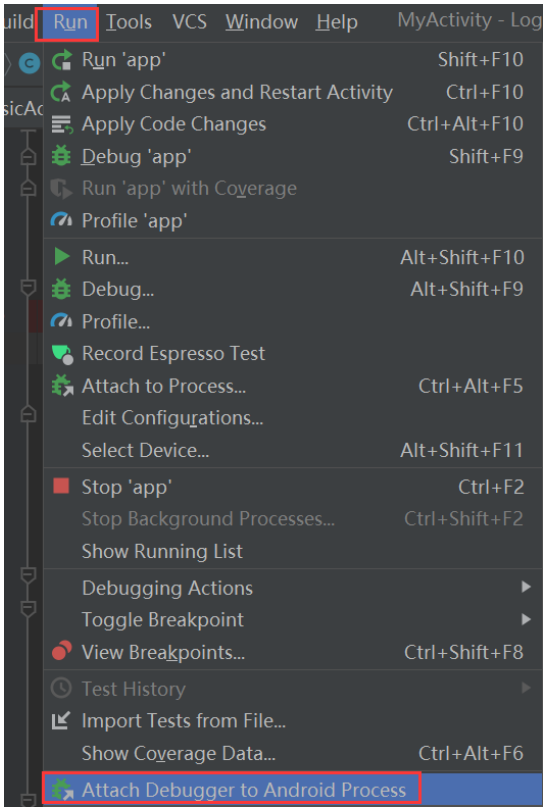


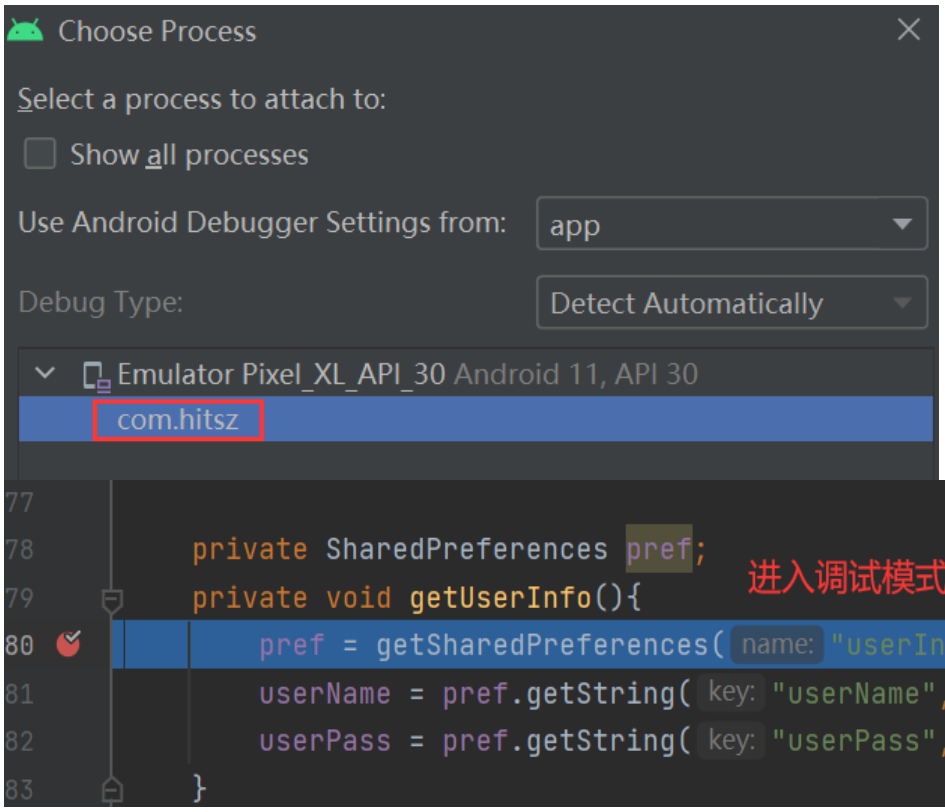
断点调试（2）

程序运行之后，开启调试模式

运行程序，点击Run->Attach debugger to Android process。attach process到指定进程，条件触发之后就可以直接进入调试模式。







基本调试流程

日志工具

✓ 日志打印，五种Log类型

Log.v: VERBOSE级别，打印最为繁琐、意义不大的日志信息；

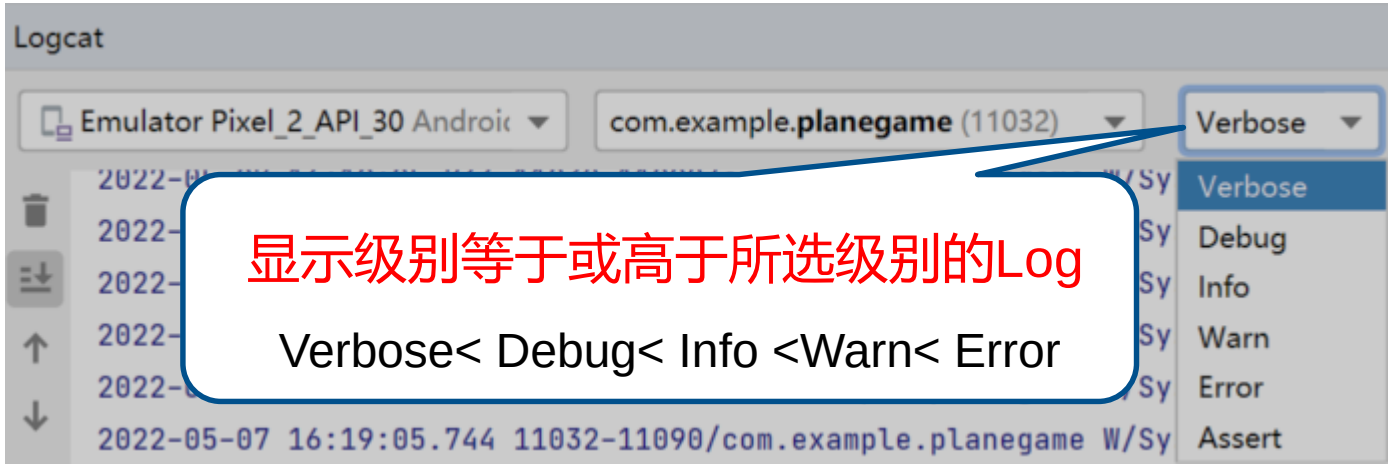
Log.d: DEBUG级别，打印调试信息；

Log.i: INFO级别，打印比较重要的数据，可以帮助分析用户行为；

Log.w：WARN级别，打印警告信息；

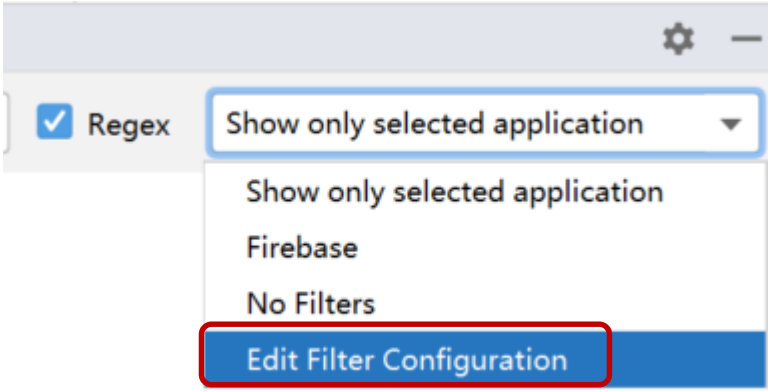
Log.e: ERROR级别，打印错误信息。

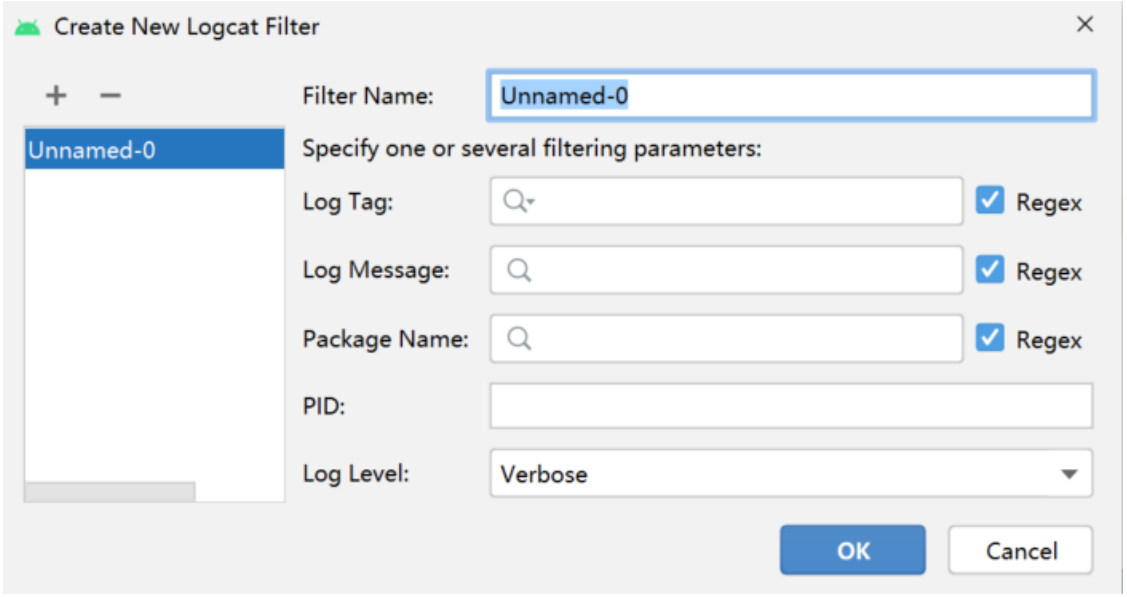
✓ System.out会重定向到Log中，级别为INFO



日志工具

✓ 设置过滤器过滤Log信息





空指针

```
2021-12-28 10:52:37.001 11020-11020/com.hitsz E/AndroidRuntime: FATAL EXCEPTION: main
    Process: com.hitsz, PID: 11020
    java.lang.RuntimeException: Unable to start activity ComponentInfo{com.hitsz/com.hitsz.activity.ListActivity}: java.lang.NullPointerException
        at android.app.ActivityThread.performLaunchActivity(ActivityThread.java:3449)
        at android.app.ActivityThread.handleLaunchActivity(ActivityThread.java:3601)
        at android.app.servertransaction.LaunchActivityItem.execute(LaunchActivityItem.java:85)
        at android.app.servertransaction.TransactionExecutor.executeCallbacks(TransactionExecutor.java:135)
        at android.app.servertransaction.TransactionExecutor.execute(TransactionExecutor.java:95)
        at android.app.ActivityThread$H.handleMessage(ActivityThread.java:2066)
        at android.os.Handler.dispatchMessage(Handler.java:106)
        at android.os.Looper.loop(Looper.java:223)
        at android.app.ActivityThread.main(ActivityThread.java:7656) <1 internal call>
        at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:592)
        at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:947)
    Caused by: java.lang.NullPointerException: Attempt to invoke virtual method 'int java.util.ArrayList.size()' on a null object reference
        at com.hitsz.view.ListAdapter.getCount(ListAdapter.java:33)
        at android.widget.ListView.setAdapter(ListView.java:581)
        at com.hitsz.activity.ListActivity.onCreate(ListActivity.java:28)
```

空指针

出错提示代码：

```
public class ListAdapter extends BaseAdapter {
    private Context mContext;
    private LayoutInflater mLayoutInflater;
    private ArrayList<ListItemData> itemDataList;
    public ListAdapter(Context context){
        this.mContext = context;
        mLayoutInflater = LayoutInflater.from(context);
    }
    @Override
    public Object getItem(int position) {
        return itemDataList.get(position);
    }
    @Override
    public int getCount() {
        return itemDataList.size();
    }
```

报错提示行

出错原因ListActivity中：

```
private ArrayList<ListItemData> itemDataList = new ArrayList<>();
private void initListItemArray(){
    for(int i = 0;i<20;i++){
        ListItemData item = new ListItemData();
        item.setTitle("标题" + i);
        item.setTime("2021-12-2" + i);
        item.setContent("内容" + i);

        Random rdm = new Random();
        item.setImage(rdm.nextInt( bound: 4));

        itemDataList.add(item);
    }
}
```

修改错误代码

温馨提示：对象在使用前做判断处理！！！

某些不影响主业务逻辑的场景即使出现空指针，但有做判断处理，不至于导致程序崩溃；

```
@Override
public int getCount() {
    if (itemDataList != null){
        return itemDataList.size();
    }
    else
        return 0;
}
```

同学们
请开始实验吧！