**package** com.ChildHealthDiet.app2;  
**import** android.content.Context;  
**import** android.util.Log;  
**import** com.ChildHealthDiet.app2.model.bean.DaoMaster;  
**import** com.ChildHealthDiet.app2.model.bean.DaoSession;  
**import** com.ChildHealthDiet.app2.utils.FileUtils;  
**import** org.greenrobot.greendao.database.Database;  
  
**public class** RecipeApplication **extends** android.app.Application {  
  
 **public** DaoSession **daoSession**;  
 **private static** RecipeApplication *instance*;  
 **public** Boolean **isCollectMode** = **false**;  
 @Override  
 **public void** onCreate() {  
 **super**.onCreate();  
 *//复制assets目录下的数据库文件到应用数据库中* **try** {  
 FileUtils.*copyDataBase*(**"db/babydietfood.db"**,**this**);  
 } **catch** (Exception e) {  
 Log.*e*(**"Application"**, e.getMessage());  
 }  
 DaoMaster.DevOpenHelper helper = **new** DaoMaster.DevOpenHelper(**this**, **"babydietfood.db"**);  
 Database db = helper.getEncryptedWritableDb(**"key123"**);  
 **daoSession** = **new** DaoMaster(db).newSession();  
 *instance* = **this**;  
 }  
 *// 获取Application* **public static** RecipeApplication getApplication() {  
 **return** *instance*;  
 }  
 **public static** Context getContext() {  
 **return** *instance*.getApplicationContext();  
 }  
}

**package** com.ChildHealthDiet.app2;  
**import** android.os.Bundle;  
**import** android.support.annotation.NonNull;  
**import** android.support.design.widget.BottomNavigationView;  
**import** android.support.v4.view.ViewPager;  
**import** android.view.MenuItem;  
**import** android.view.MotionEvent;  
**import** android.view.View;  
**import** com.ChildHealthDiet.app2.adapter.ViewPagerAdapter;  
**import** com.ChildHealthDiet.app2.event.ChangeTabEvent;  
**import** com.ChildHealthDiet.app2.ui.base.BaseActivity;  
**import** com.ChildHealthDiet.app2.ui.fragment.CategoryFragment;  
**import** com.ChildHealthDiet.app2.ui.fragment.HomeFragment;  
**import** com.ChildHealthDiet.app2.ui.fragment.MineFragment;  
**import** com.ChildHealthDiet.app2.utils.BottomNavigationViewHelper;  
**import** butterknife.BindView;  
**import** io.reactivex.android.schedulers.AndroidSchedulers;  
**import** io.reactivex.disposables.Disposable;  
**public class** MainActivity **extends** BaseActivity {  
  
 @BindView(R.id.***viewpager***) ViewPager **viewPager**; MenuItem **menuItem**;  
 @BindView(R.id.***bottom\_navigation***)  
 BottomNavigationView **bottomNavigationView**;  
  
 @Override  
 **protected int** getContentId() {  
 **return** R.layout.***activity\_main***;  
 }  
 @Override  
 **protected void** initWidget() {  
 BottomNavigationViewHelper.*disableShiftMode*(**bottomNavigationView**);  
 BottomNavigationViewHelper.*disableItemScale*(**bottomNavigationView**);  
 **bottomNavigationView**.setOnNavigationItemSelectedListener(  
 **new** BottomNavigationView.OnNavigationItemSelectedListener() {  
 @Override  
 **public boolean** onNavigationItemSelected(@NonNull MenuItem item) {  
 **switch** (item.getItemId()) {  
 **case** R.id.***navigation\_home***:  
 **viewPager**.setCurrentItem(0);  
 **break**;  
 **case** R.id.***navigation\_category***:  
 **viewPager**.setCurrentItem(1);  
 **break**;  
 **case** R.id.***navigation\_mine***:  
 **viewPager**.setCurrentItem(2);  
 **break**;}  
 **return false**;  
 }  
 });  
 **viewPager**.addOnPageChangeListener(**new** ViewPager.OnPageChangeListener() {  
 @Override  
 **public void** onPageScrolled(**int** position, **float** positionOffset, **int** positionOffsetPixels) {  
 }  
 @Override  
 **public void** onPageSelected(**int** position) {  
 **if** (**menuItem** != **null**) {  
 **menuItem**.setChecked(**false**);  
 } **else** {  
 **bottomNavigationView**.getMenu().getItem(0).setChecked(**false**);  
 }  
 **menuItem** = **bottomNavigationView**.getMenu().getItem(position);  
 **menuItem**.setChecked(**true**);  
 }  
 @Override  
 **public void** onPageScrollStateChanged(**int** state) {  
 }  
 });  
 *//禁止ViewPager滑动* **viewPager**.setOnTouchListener(**new** View.OnTouchListener() {  
 @Override  
 **public boolean** onTouch(View v, MotionEvent event) {  
 **return true**;  
 }  
 });  
 setupViewPager(**viewPager**);  
 initEvent();  
  
 }  
  
 **private void** initEvent(){  
 Disposable tabDisp = RxBus.*getInstance*()  
 .toObservable(ChangeTabEvent.**class**)  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(  
 event -> {

MainActivity.**this**.**viewPager**.setCurrentItem(event.**tabIndex**);  
 }  
 );  
 addDisposable(tabDisp);  
 }  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 *//默认 >3 的选中效果会影响ViewPager的滑动切换时的效果，故利用反射去掉* }  
  
 **private void** setupViewPager(ViewPager viewPager) {  
 ViewPagerAdapter adapter = **new** ViewPagerAdapter(getSupportFragmentManager());  
 adapter.addFragment(**new** HomeFragment());  
 adapter.addFragment(**new** CategoryFragment());  
 adapter.addFragment(**new** MineFragment());  
  
 viewPager.setOffscreenPageLimit(3);  
 viewPager.setAdapter(adapter);  
  
 }  
  
}

**package** com.ChildHealthDiet.app2.adapter;  
**import** android.content.Context;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.widget.TextView;  
**import** com.ChildHealthDiet.app2.R;  
**import** com.ChildHealthDiet.app2.model.bean.KeyValueBean;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeCategory;  
**import** com.ChildHealthDiet.app2.ui.activitys.RecipeKeywordListActivity;  
**import** com.ChildHealthDiet.app2.ui.base.SuperViewHolder;  
**import** com.ChildHealthDiet.app2.ui.base.adapter.ListBaseAdapter;  
**import** com.ChildHealthDiet.app2.ui.categorys.RECIPETYPE;  
**import** com.zhy.view.flowlayout.FlowLayout;  
**import** com.zhy.view.flowlayout.TagAdapter;  
**import** com.zhy.view.flowlayout.TagFlowLayout;  
**import** java.util.Set;  
**public class** CategoryRecipeDataAdapter **extends** ListBaseAdapter<RecipeCategory> {  
  
 **public** CategoryRecipeDataAdapter(Context context) {  
 **super**(context);  
 }  
 @Override  
 **public int** getLayoutId() {  
 **return** R.layout.***category\_recipe\_item\_layout***;  
 }  
  
 @Override  
 **public void** onBindItemHolder(SuperViewHolder holder, **int** position) {  
 RecipeCategory item = **mDataList**.get(position);  
  
 TextView titleText = holder.getView(R.id.***id\_categorys\_name***);  
 titleText.setText(item.**name**);  
  
 **final** LayoutInflater mInflater = LayoutInflater.*from*(**mContext**);  
 TagFlowLayout mFlowLayout = (TagFlowLayout) holder.getView(R.id.***id\_categorys\_flowlayout***);  
 mFlowLayout.setAdapter(**new** TagAdapter<RecipeCategory.CategoryName>(item.**categorys**)  
 {  
 @Override  
 **public** View getView(FlowLayout parent, **int** position, RecipeCategory.CategoryName s)  
 {  
 TextView tv = (TextView) mInflater.inflate(R.layout.***tag\_tv***,  
 mFlowLayout, **false**);  
 tv.setText(s.**name**);  
 **return** tv;  
 }  
  
 @Override  
 **public boolean** setSelected(**int** position, RecipeCategory.CategoryName s)  
 {  
 **return false**;  
 }  
 });  
  
 mFlowLayout.setOnTagClickListener(**new** TagFlowLayout.OnTagClickListener()  
 {  
 @Override  
 **public boolean** onTagClick(View view, **int** p, FlowLayout parent)  
 {String strKey = item.**categorys**.get(p).**key**;  
 **switch** (position){  
 **case** 0:  
 RecipeKeywordListActivity.*startActivity*(**mContext**,RECIPETYPE.***Eattime***,strKey);  
 **break**;  
 **case** 1:  
 KeyValueBean keyValueBean = **new** KeyValueBean(strKey,item.**categorys**.get(p).**name**+**"宝宝食谱"**);  
 RecipeKeywordListActivity.*startActivity*(**mContext**,RECIPETYPE.***Month***,keyValueBean);  
 **break**;  
 **case** 2:  
 **case** 3:  
 RecipeKeywordListActivity.*startActivity*(**mContext**,RECIPETYPE.***Symptoms***,strKey);  
 **break**;

**case** 4:  
 **case** 5:  
 **case** 6:  
 RecipeKeywordListActivity.*startActivity*(**mContext**,RECIPETYPE.***Ingredients***,strKey);  
 **break**;  
 }  
 **return true**;  
 }  
 });  
 mFlowLayout.setOnSelectListener(**new** TagFlowLayout.OnSelectListener()  
 {  
 @Override  
 **public void** onSelected(Set<Integer> selectPosSet)  
 {  
 }  
 });  
 }  
 @Override  
 **public void** onViewRecycled(SuperViewHolder holder) {  
  
 **super**.onViewRecycled(holder);  
 }  
  
}

**package** com.ChildHealthDiet.app2.adapter;  
**import** android.content.Context;  
**import** android.view.View;  
**import** android.widget.ImageView;  
**import** android.widget.TextView;  
  
**import** com.bumptech.glide.Glide;  
**import** com.ChildHealthDiet.app2.R;  
**import** com.ChildHealthDiet.app2.model.bean.MineItem;  
**import** com.ChildHealthDiet.app2.ui.base.SuperViewHolder;  
**import** com.ChildHealthDiet.app2.ui.base.adapter.ListBaseAdapter;  
  
**public class** MineDataAdapter **extends** ListBaseAdapter<MineItem> {  
  
 **public** MineDataAdapter(Context context) {  
 **super**(context);  
 }  
  
 @Override  
 **public int** getLayoutId() {  
 **return** R.layout.***mine\_item\_layout***;  
 }  
  
 @Override  
 **public void** onBindItemHolder(SuperViewHolder holder, **int** position) {  
  
 MineItem item = **mDataList**.get(position);  
  
 TextView titleText = holder.getView(R.id.***tv\_item\_name***);  
 titleText.setText(item.getName());  
  
 ImageView imvitem = holder.getView(R.id.***mineitem\_imv***);  
 Glide.*with*(**mContext**)  
 .load(item.getPicture())  
 .into(imvitem);  
 View lineView = holder.getView(R.id.***mine\_item\_line***);  
 **if**(position == **mDataList**.size()-1){  
 lineView.setVisibility(View.***INVISIBLE***);  
 }  
 }  
 @Override  
 **public void** onViewRecycled(SuperViewHolder holder) {  
 **super**.onViewRecycled(holder);  
 }  
}

**package** com.ChildHealthDiet.app2.adapter;  
  
**import** android.content.Context;  
**import** android.widget.TextView;  
**import** com.bumptech.glide.Glide;  
**import** com.ChildHealthDiet.app2.R;  
**import** com.ChildHealthDiet.app2.model.bean.MonthRecipe;  
**import** com.ChildHealthDiet.app2.ui.base.SuperViewHolder;  
**import** com.ChildHealthDiet.app2.ui.base.adapter.ListBaseAdapter;  
**import** com.makeramen.roundedimageview.RoundedImageView;  
  
**public class** MonthRecipeDataAdapter **extends** ListBaseAdapter<MonthRecipe> {  
  
 **public** MonthRecipeDataAdapter(Context context) {  
 **super**(context);  
 }  
  
 @Override  
 **public int** getLayoutId() {  
 **return** R.layout.***month\_recipe\_item\_layout***;  
 }  
  
 @Override  
 **public void** onBindItemHolder(SuperViewHolder holder, **int** position) {  
 MonthRecipe item = **mDataList**.get(position);  
  
 TextView titleText = holder.getView(R.id.***recipeName***);  
 titleText.setText(item.**title**);  
 TextView numberText = holder.getView(R.id.***recipeNumber***);  
 numberText.setText(item.**subTitle**+**"篇菜谱"**);  
 RoundedImageView iv1 = holder.getView(R.id.***iv\_month\_recipe\_item1***);  
 RoundedImageView iv2 = holder.getView(R.id.***iv\_month\_recipe\_item2***);  
 RoundedImageView iv3 = holder.getView(R.id.***iv\_month\_recipe\_item3***);  
 RoundedImageView iv4 = holder.getView(R.id.***iv\_month\_recipe\_item4***);  
 Glide.*with*(**mContext**)  
 .load(**"file:///android\_asset/recipeimage/"**+item.**images**.get(0))  
 .into(iv1);  
 Glide.*with*(**mContext**)  
 .load(**"file:///android\_asset/recipeimage/"**+item.**images**.get(1))  
 .into(iv2);  
 Glide.*with*(**mContext**)  
 .load(**"file:///android\_asset/recipeimage/"**+item.**images**.get(2))  
 .into(iv3);  
 Glide.*with*(**mContext**)  
 .load(**"file:///android\_asset/recipeimage/"**+item.**images**.get(3))  
 .into(iv4);  
 }  
  
 @Override  
 **public void** onViewRecycled(SuperViewHolder holder) {  
  
 **super**.onViewRecycled(holder);  
 }  
  
}

**package** com.ChildHealthDiet.app2.adapter;  
**import** android.content.Context;  
**import** android.widget.TextView;  
**import** com.ChildHealthDiet.app2.R;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeBean;  
**import** com.ChildHealthDiet.app2.ui.base.SuperViewHolder;  
**import** com.ChildHealthDiet.app2.ui.base.adapter.ListBaseAdapter;  
  
**public class** RecipeBasketDataAdapter **extends** ListBaseAdapter<RecipeBean> {  
  
 **public** RecipeBasketDataAdapter(Context context) {  
 **super**(context);  
 }  
 @Override  
 **public int** getLayoutId() {  
 **return** R.layout.***basket\_item\_layout***;  
 }  
 @Override  
 **public void** onBindItemHolder(SuperViewHolder holder, **int** position) {  
 RecipeBean item = **mDataList**.get(position);  
 TextView titleText = holder.getView(R.id.***tv\_basket\_repice\_name***);  
 titleText.setText(item.getName());  
 TextView tvContent = holder.getView(R.id.***tv\_basket\_repice\_ingredients***);  
 tvContent.setText(item.getIngredients().replaceAll(**"<br>"**,**"\r\n"**));  
 TextView tvMonth = holder.getView(R.id.***tv\_basket\_repice\_month***);  
 tvMonth.setText(item.getMonth()+**"个月"**);  
 }  
 @Override  
 **public void** onViewRecycled(SuperViewHolder holder) {  
 **super**.onViewRecycled(holder);  
 }  
}

**package** com.ChildHealthDiet.app2.adapter;  
**import** android.content.Context;  
**import** android.support.v7.widget.RecyclerView;  
**import** android.text.TextUtils;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.Button;  
**import** android.widget.ImageButton;  
**import** android.widget.ImageView;  
**import** android.widget.RelativeLayout;  
**import** android.widget.TextView;  
**import** com.bumptech.glide.Glide;  
**import** com.ChildHealthDiet.app2.R;  
**import** com.ChildHealthDiet.app2.model.bean.CookRecipeMethod;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeBean;  
**import** com.ChildHealthDiet.app2.presenter.contract.RecipeDetailContract;  
**import** com.ChildHealthDiet.app2.ui.components.Utils;  
**import** java.util.ArrayList;  
**import** java.util.List;  
**import** butterknife.BindView;  
**import** butterknife.ButterKnife;  
**public class** RecipeDetailAdapter **extends** RecyclerView.Adapter<RecipeDetailAdapter.ItemViewHolder>{  
  
 **private** Context **context**;  
 **public** RecipeBean **srcData**;  
 **private** List<CookDetailStruct> **datas**;  
 **private** String **sumary**;  
 **private** ArrayList<String> **ingredientsDatas**;  
 **private** ArrayList<CookRecipeMethod> **cookRecipeMethods**;  
 **private boolean isShowCollection**;  
 **private** RecipeDetailContract.Presenter **mRecipedetailpresenter**;  
**public** RecipeDetailAdapter(Context context, RecipeBean data, RecipeDetailContract.Presenter recipedetailpresenter, **boolean** isShowCollection) {  
 **this**.**context** = context;  
 **this**.**srcData** = data;  
 **this**.**isShowCollection** = isShowCollection;  
 **this**.**mRecipedetailpresenter** = recipedetailpresenter;  
  
 **sumary** = data.getPrompt();  
 **sumary** = **sumary**.replaceAll(**"<br>"**,**""**);  
  
 String str = data.getIngredients();  
 **if**(**null** == str || TextUtils.*isEmpty*(str)) {  
 **ingredientsDatas** = **new** ArrayList<>();  
 }  
 **else**{  
 String[] ingredients = str.split(**"<br>"**);  
  
 **ingredientsDatas** = **new** ArrayList<String>();**for**(**int** i=0;i<ingredients.**length**;i++) {  
  
 **ingredientsDatas**.add(ingredients[i]);  
 }  
 }  
  
 str = data.getPractice();  
 **if**(**null** == str || TextUtils.*isEmpty*(str)){  
 **cookRecipeMethods** = **new** ArrayList<>();  
 }  
 **else** {  
 String[] steps = **null**;  
 **cookRecipeMethods** = **new** ArrayList<>();  
 **if**(Utils.*isMatchregEx*(str,**"[1-9]、"**)) {  
 steps = str.split(**"[1-9]、"**);  
 }  
 **else if**(Utils.*isMatchregEx*(str,**"<br/>"**)){  
  
 }  
 **else** {  
 steps = **new** String[]{str};  
 }  
 **int** stepIndex = 1;  
 **for**(**int** i=0;i<steps.**length**;i++) {  
 String stepItem = steps[i].trim();  
 **if**(stepItem.equals(**""**)){  
 **continue**;  
 }  
 CookRecipeMethod cookRecipeMethod = **new** CookRecipeMethod();  
 cookRecipeMethod.setStep( stepIndex+**". "**+stepItem);  
 **cookRecipeMethods**.add(cookRecipeMethod);  
 stepIndex++;  
 }  
 }  
 **this**.**datas** = CookDetailStruct.*create*(**cookRecipeMethods**);  
 }  
  
 @Override  
 **public** ItemViewHolder onCreateViewHolder(ViewGroup parent, **int** viewType){  
 **if**(***Cook\_Detail\_Item\_Type\_CookMan*** == viewType){  
 CookManItemViewHolder holder = **new** CookManItemViewHolder(  
 LayoutInflater.*from*(**context**).inflate(R.layout.***item\_cook\_detail\_cookman***, parent, **false**)  
 );  
 **return** holder;  
 }  
 **else if**(***Cook\_Detail\_Item\_Type\_Header*** == viewType){  
 HeaderItemViewHolder holder = **new** HeaderItemViewHolder(  
 LayoutInflater.*from*(**context**).inflate(R.layout.***item\_cook\_detail\_header***, parent, **false**)  
 );  
  
 **return** holder;  
 }  
 **else**{  
 StepItemViewHolder holder = **new** StepItemViewHolder(  
 LayoutInflater.*from*(**context**).inflate(R.layout.***item\_cook\_detail\_step***, parent, **false**)  
 );  
 **return** holder;  
 }  
 }  
 @Override  
 **public void** onBindViewHolder(ItemViewHolder holder, **int** position){  
  
 **if**(***Cook\_Detail\_Item\_Type\_CookMan*** == getItemViewType(position)){  
 **final** CookManItemViewHolder holderView = (CookManItemViewHolder)holder;  
 holderView.**textSumary**.setText(**sumary**);  
 **return** ;  
 }  
 **if**(***Cook\_Detail\_Item\_Type\_Header*** == getItemViewType(position)){  
 HeaderItemViewHolder holderView = (HeaderItemViewHolder)holder;  
 **if**(**ingredientsDatas**.size() < 1){  
 holderView.**view1**.setVisibility(View.***GONE***);  
 holderView.**view2**.setVisibility(View.***GONE***);  
 holderView.**view3**.setVisibility(View.***GONE***);  
 holderView.**textIngredients**.setVisibility(View.***GONE***);  
 }  
 **else if**(**ingredientsDatas**.size() < 2) {  
 holderView.**view1**.setVisibility(View.***VISIBLE***);  
 holderView.**view2**.setVisibility(View.***GONE***);  
 holderView.**view3**.setVisibility(View.***GONE***); holderView.**textIngredientsContent1**.setText(**ingredientsDatas**.get(0));  
 }  
 **else if**(**ingredientsDatas**.size() < 3){  
 holderView.**view1**.setVisibility(View.***VISIBLE***);  
 holderView.**view2**.setVisibility(View.***VISIBLE***);  
 holderView.**view3**.setVisibility(View.***GONE***); holderView.**textIngredientsContent1**.setText(**ingredientsDatas**.get(0));  
 holderView.**textIngredientsContent2**.setText(**ingredientsDatas**.get(1));  
 }  
 **else**{  
 holderView.**view1**.setVisibility(View.***VISIBLE***);  
 holderView.**view2**.setVisibility(View.***VISIBLE***);  
 holderView.**view3**.setVisibility(View.***VISIBLE***);  
 holderView.**textIngredientsContent1**.setText(**ingredientsDatas**.get(0)); holderView.**textIngredientsContent2**.setText(**ingredientsDatas**.get(1));  
 holderView.**textIngredientsContent3**.setText(**ingredientsDatas**.get(2));  
 }  
 **if**(**srcData**.getMonth() != **null**){  
 holderView.**tvSubTitle**.setText(**srcData**.getMonth()+**"个月宝宝食谱"**);  
 }  
 **else if**(**srcData**.getType() != **null**){  
 holderView.**tvSubTitle**.setText(**srcData**.getType());  
 }  
 **if**(**srcData**.getCollect())  
 {  
 holderView.**btn\_collect**.setBackgroundColor(**context**.getResources().getColor(R.color.***colorPrimaryGray***));  
 holderView.**btn\_collect**.setText(**"已收藏"**);  
 holderView.**btn\_collect**.setTextColor(**context**.getResources().getColor(R.color.***black\_text***));  
 }  
 **else** { holderView.**btn\_collect**.setBackgroundColor(**context**.getResources().getColor(R.color.***colorPrimaryDark***));  
 holderView.**btn\_collect**.setText(**"收藏"**);  
 holderView.**btn\_collect**.setTextColor(**context**.getResources().getColor(R.color.***lightgray***));  
 }  
  
 **if**(**srcData**.getBasket()) { holderView.**btn\_shopping\_cart**.setBackgroundColor(**context**.getResources().getColor(R.color.***colorPrimaryGray***));  
 holderView.**btn\_shopping\_cart**.setText(**"已加入购物清单"**);  
 holderView.**btn\_shopping\_cart**.setTextColor(**context**.getResources().getColor(R.color.***black\_text***));  
 }  
 **else** {  
 holderView.**btn\_shopping\_cart**.setBackgroundColor(**context**.getResources().getColor(R.color.***colorPrimaryDark***));  
 holderView.**btn\_shopping\_cart**.setText(**"加入购物清单"**); holderView.**btn\_shopping\_cart**.setTextColor(**context**.getResources().getColor(R.color.***lightgray***));  
 }  
   
 holderView.**btn\_collect**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **srcData**.setCollect(!**srcData**.getCollect());  
 notifyDataSetChanged();  
 **mRecipedetailpresenter**.updateRecipeBeanData(RecipeDetailAdapter.**this**.**context**,**srcData**);  
 }  
 });  
 holderView.**btn\_shopping\_cart**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **srcData**.setBasket(!**srcData**.getBasket());  
 notifyDataSetChanged(); **mRecipedetailpresenter**.updateRecipeBeanData(RecipeDetailAdapter.**this**.**context**,**srcData**);  
 }  
 });  
 **return** ;  
 }  
 StepItemViewHolder holderView = (StepItemViewHolder)holder;  
 CookRecipeMethod data = **datas**.get(position).getData();  
 holderView.**textContent**.setText(data.getStep());  
 **if**(data.getImg() != **null** && (!TextUtils.*isEmpty*(data.getImg()))) {  
 holderView.**imgvStep**.setVisibility(View.***VISIBLE***);  
 Glide.*with*(**context**)  
 .load(data.getImg())  
 .into(holderView.**imgvStep**);  
 }  
 **else**{  
 holderView.**imgvStep**.setVisibility(View.***GONE***);  
 }  
 }  
 @Override  
 **public int** getItemCount(){  
 **return datas**.size();  
 }  
  
 @Override  
 **public int** getItemViewType(**int** position) {  
 **return datas**.get(position).getType();  
 }  
  
 **private final static int *Cook\_Detail\_Item\_Type\_CookMan*** = 0;  
 **private final static int *Cook\_Detail\_Item\_Type\_Header*** = 1;  
 **private final static int *Cook\_Detail\_Item\_Type\_Step*** = 2;  
 **private static class** CookDetailStruct{  
 **private int type**;  
 **private** CookRecipeMethod **data**;

**public** CookDetailStruct(**int** type){  
 **this**.**type** = type;  
 }  
 **public** CookDetailStruct(**int** type, CookRecipeMethod data){  
 **this**.**type** = type;  
 **this**.**data** = data;  
 }  
 **public int** getType() {  
 **return type**;  
 }  
  
 **public** CookRecipeMethod getData() {  
 **return data**;  
 }  
  
 **public static** List<CookDetailStruct> create(ArrayList<CookRecipeMethod> cookRecipeMethods){  
 List<CookDetailStruct> datas = **new** ArrayList<>();  
 datas.add(**new** CookDetailStruct(***Cook\_Detail\_Item\_Type\_Header***));  
 **if**(**null** == cookRecipeMethods)  
 **return** datas;  
 **for**(CookRecipeMethod item : cookRecipeMethods) {  
 datas.add(**new** CookDetailStruct(***Cook\_Detail\_Item\_Type\_Step***, item));  
 }  
 datas.add(**new** CookDetailStruct(***Cook\_Detail\_Item\_Type\_CookMan***));  
 **return** datas;  
 }  
 }  
  
 **public class** StepItemViewHolder **extends** ItemViewHolder{  
  
 @BindView(R.id.***text\_content***)  
 **public** TextView **textContent**;  
 @BindView(R.id.***imgv\_step***)  
 **public** ImageView **imgvStep**;  
 **public** StepItemViewHolder(View itemView){  
 **super**(itemView);  
 }  
 }  
  
 **public class** CookManItemViewHolder **extends** ItemViewHolder{  
 @BindView(R.id.***text\_sumary***)  
 **public** TextView **textSumary**;  
 **public** CookManItemViewHolder(View itemView){  
 **super**(itemView);  
 }  
 }  
  
 **public class** HeaderItemViewHolder **extends** ItemViewHolder{  
  
 @BindView(R.id.***text\_subTitle***)  
 **public** TextView **tvSubTitle**;  
 @BindView(R.id.***text\_ingredients***)  
 **public** TextView **textIngredients**;  
 @BindView(R.id.***relative\_view1***)  
 **public** RelativeLayout **view1**;  
 @BindView(R.id.***relative\_view2***)  
 **public** RelativeLayout **view2**;  
 @BindView(R.id.***relative\_view3***)  
 **public** RelativeLayout **view3**;  
 @BindView(R.id.***text\_ingredients\_content1***)  
 **public** TextView **textIngredientsContent1**;  
 @BindView(R.id.***text\_ingredients\_content2***)  
 **public** TextView **textIngredientsContent2**;  
 @BindView(R.id.***text\_ingredients\_content3***)  
 **public** TextView **textIngredientsContent3**;  
 @BindView(R.id.***recipe\_collect***)  
 **public** ImageButton **imgButtonCollect**;  
  
 @BindView(R.id.***btn\_collect***)  
 **public** Button **btn\_collect**;  
 @BindView(R.id.***btn\_shopping\_cart***)  
 **public** Button **btn\_shopping\_cart**;  
  
 **public** HeaderItemViewHolder(View itemView){  
 **super**(itemView);  
 }  
 }  
  
 **public class** ItemViewHolder **extends** RecyclerView.ViewHolder{  
  
 **public** ItemViewHolder(View itemView){  
 **super**(itemView);  
 ButterKnife.*bind*(**this**, itemView);  
 }  
 }  
}

**package** com.ChildHealthDiet.app2.adapter;  
  
**import** android.content.Context;  
**import** android.view.View;  
**import** android.widget.CheckBox;  
**import** android.widget.ImageView;  
**import** android.widget.TextView;  
**import** com.bumptech.glide.Glide;  
**import** com.ChildHealthDiet.app2.R;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeBean;  
**import** com.ChildHealthDiet.app2.ui.base.SuperViewHolder;  
**import** com.ChildHealthDiet.app2.ui.base.adapter.ListBaseAdapter;  
  
**public class** RecipeListDataAdapter **extends** ListBaseAdapter<RecipeBean> {  
  
 **public** RecipeListDataAdapter(Context context) {  
 **super**(context);  
 }  
 @Override  
 **public int** getLayoutId() {  
 **return** R.layout.***recipe\_item\_layout***;  
 }  
 @Override  
 **public void** onBindItemHolder(SuperViewHolder holder, **int** position) {  
 RecipeBean item = **mDataList**.get(position);  
 TextView titleText = holder.getView(R.id.***tv\_title***);  
 titleText.setText(item.getName());  
 TextView tvMonth = holder.getView(R.id.***tv\_month***);  
 String strMonth = item.getMonth() != **null** ? item.getMonth()+**"个月"**:(item.getSymptoms() != **null**?item.getSymptoms():**""**);  
 tvMonth.setText(strMonth);  
  
 TextView tvContent = holder.getView(R.id.***tv\_content***);  
 tvContent.setText(item.getPrompt());  
  
 ImageView imvRecipe = holder.getView(R.id.***imv\_recipe***);  
 Glide.*with*(**mContext**)  
 .load(**"file:///android\_asset/recipeimage/"**+item.getPicture())  
 .into(imvRecipe);  
 CheckBox chk = holder.getView(R.id.***chkRecipeItem***);  
 chk.setVisibility(item.getShowCheckBox()?View.***VISIBLE***:View.***GONE***);  
 chk.setChecked(item.**isCanDelete**);  
 chk.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View arg0) {  
 item.**isCanDelete** = chk.isChecked();  
 }  
 });  
 }  
  
 @Override  
 **public void** onViewRecycled(SuperViewHolder holder) {  
 **super**.onViewRecycled(holder);  
 }  
  
}

**package** com.ChildHealthDiet.app2.adapter;  
  
**import** android.support.v4.app.Fragment;  
**import** android.support.v4.app.FragmentManager;  
**import** android.support.v4.app.FragmentPagerAdapter;  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** ViewPagerAdapter **extends** FragmentPagerAdapter {  
  
 **private final** List<Fragment> **mFragmentList** = **new** ArrayList<>();  
 **public** ViewPagerAdapter(FragmentManager manager) {  
 **super**(manager);  
 }  
 @Override  
 **public** Fragment getItem(**int** position) {  
 **return mFragmentList**.get(position);  
 }  
 @Override  
 **public int** getCount() {  
 **return mFragmentList**.size();  
 }  
 **public void** addFragment(Fragment fragment) {  
 **mFragmentList**.add(fragment);  
 }  
}

**package** com.ChildHealthDiet.app2.utils;  
**import** java.util.Calendar;  
**import** java.util.Date;  
**import** java.util.GregorianCalendar;  
  
**public class** AgeUtils {  
 *// 根据年月日计算年龄,birthTimeString:"1994-11-14"* **public static int** getAgeFromBirthTime(String birthTimeString) {  
 *// 先截取到字符串中的年、月、日* String strs[] = birthTimeString.trim().split(**"-"**);  
 **int** selectYear = Integer.*parseInt*(strs[0]);  
 **int** selectMonth = Integer.*parseInt*(strs[1]);  
 **int** selectDay = Integer.*parseInt*(strs[2]);  
 *// 得到当前时间的年、月、日* Calendar cal = Calendar.*getInstance*();  
 **int** yearNow = cal.get(Calendar.***YEAR***);  
 **int** monthNow = cal.get(Calendar.***MONTH***) + 1;  
 **int** dayNow = cal.get(Calendar.***DATE***);  
 *// 用当前年月日减去生日年月日* **int** yearMinus = yearNow - selectYear;  
 **int** monthMinus = monthNow - selectMonth;  
 **int** dayMinus = dayNow - selectDay;  
 **int** age = yearMinus;*// 先大致赋值* **if** (yearMinus < 0) {*// 选了未来的年份* age = 0;  
 } **else if** (yearMinus == 0) {*// 同年的，要么为1，要么为0* **if** (monthMinus < 0) {*// 选了未来的月份* age = 0;  
 } **else if** (monthMinus == 0) {*// 同月份的* **if** (dayMinus < 0) {*// 选了未来的日期* age = 0;  
 } **else if** (dayMinus >= 0) {  
 age = 1;  
 }  
 } **else if** (monthMinus > 0) {  
 age = 1;  
 }  
 } **else if** (yearMinus > 0) {  
 **if** (monthMinus < 0) {*// 当前月>生日月* } **else if** (monthMinus == 0) {*// 同月份的，再根据日期计算年龄* **if** (dayMinus < 0) {  
 } **else if** (dayMinus >= 0) {  
 age = age + 1;  
 }  
 } **else if** (monthMinus > 0) {  
 age = age + 1;  
 }  
 }  
 **return** age;  
 }  
  
 **public static** String getAgeFromBirthTime(Date birthDay) {  
 Calendar cal = Calendar.*getInstance*();  
 *// 得到当前时间的年、月、日* **int** yearNow = cal.get(Calendar.***YEAR***);  
 **int** monthNow = cal.get(Calendar.***MONTH***)+1;  
 **int** dayNow = cal.get(Calendar.***DATE***);  
 *// 先截取到传入中的年、月、日* cal.setTime(birthDay);  
 **int** selectYear = cal.get(Calendar.***YEAR***);  
 **int** selectMonth = cal.get(Calendar.***MONTH***) + 1;  
 **int** selectDay = cal.get(Calendar.***DAY\_OF\_MONTH***);  
 *// 用当前年月日减去生日年月日* **int** yearMinus = yearNow - selectYear;  
 **int** monthMinus = monthNow - selectMonth;  
 **int** dayMinus = dayNow - selectDay;  
 String ageToMonth= **"1月"**;  
 **if** (yearMinus > 0){  
 ageToMonth = String.*valueOf*(yearMinus)+**"岁"**;  
 } **else if** (monthMinus > 0) {  
 ageToMonth = String.*valueOf*(monthMinus)+**"月"**;  
 }  
 **return** ageToMonth;  
 }  
  
 **public static int** getMonthAge(Date date){  
 Calendar cal = Calendar.*getInstance*();  
 cal.setTime(date);  
 **int** selectYear = cal.get(Calendar.***YEAR***);  
 **int** selectMonth = cal.get(Calendar.***MONTH***) + 1;  
 **int** selectDay = cal.get(Calendar.***DAY\_OF\_MONTH***);  
 Calendar birthday = **new** GregorianCalendar(selectYear, selectMonth,selectDay);*//2010年10月12日，month从0开始* Calendar now = Calendar.*getInstance*();  
 **int** day = now.get(Calendar.***DAY\_OF\_MONTH***) - birthday.get(Calendar.***DAY\_OF\_MONTH***);  
 **int** month = now.get(Calendar.***MONTH***)+1 - birthday.get(Calendar.***MONTH***);  
 **int** year = now.get(Calendar.***YEAR***) - birthday.get(Calendar.***YEAR***);  
 *//按照减法原理，先day相减，不够向month借；然后month相减，不够向year借；最后year相减。* **if**(day<0){  
 month -= 1;  
 now.add(Calendar.***MONTH***, -1);*//得到上一个月，用来得到上个月的天数。* day = day + now.getActualMaximum(Calendar.***DAY\_OF\_MONTH***);  
 }  
 **if**(month<0){  
 month = (month+12)%12;  
 year--;  
 }  
 **if**(year>0){  
 month = month+year\*12;  
 }  
 **return** (month);  
 }  
  
 **public static** String getAge(Date date){  
 Calendar cal = Calendar.*getInstance*();  
 cal.setTime(date);  
 **int** selectYear = cal.get(Calendar.***YEAR***);  
 **int** selectMonth = cal.get(Calendar.***MONTH***) + 1;  
 **int** selectDay = cal.get(Calendar.***DAY\_OF\_MONTH***);  
 Calendar birthday = **new** GregorianCalendar(selectYear, selectMonth,selectDay);*//2010年10月12日，month从0开始* Calendar now = Calendar.*getInstance*();  
 **int** day = now.get(Calendar.***DAY\_OF\_MONTH***) - birthday.get(Calendar.***DAY\_OF\_MONTH***);  
 **int** month = now.get(Calendar.***MONTH***)+1 - birthday.get(Calendar.***MONTH***);  
 **int** year = now.get(Calendar.***YEAR***) - birthday.get(Calendar.***YEAR***);  
 *//按照减法原理，先day相减，不够向month借；然后month相减，不够向year借；最后year相减。* **if**(day<0){  
 month -= 1;  
 now.add(Calendar.***MONTH***, -1);*//得到上一个月，用来得到上个月的天数。* day = day + now.getActualMaximum(Calendar.***DAY\_OF\_MONTH***);  
 }  
 **if**(month<0){  
 month = (month+12)%12;  
 year--;  
 }  
  
 **return** ((year > 0? (year+**"岁"**) : **""** )+(month>0?(month+**"月"**):**"零"**)+day+**"天"**);  
 }  
  
}

**package** com.ChildHealthDiet.app2.context;  
**import** com.ChildHealthDiet.app2.utils.AgeUtils;  
**import** java.text.ParseException;  
**import** java.text.SimpleDateFormat;  
  
**public class** Kidinfo {  
 **private** String **nickName** = **""**;  
 **private** String **birthdate** = **""**;  
  
 **public** Kidinfo(String strNickName,String strBirthdate){  
 **this**.**nickName** = strNickName;  
 **this**.**birthdate** = strBirthdate;  
 }  
  
 **public** String getNickName() {  
 **return nickName**;  
 }  
  
 **public void** setNickName(String nickName) {  
 **this**.**nickName** = nickName;  
 }  
  
 **public** String getBirthdate() {  
 **return birthdate**;  
 }  
  
 **public void** setBirthdate(String birthdate) {  
 **this**.**birthdate** = birthdate;  
 }  
  
 **public** String getAge() **throws** ParseException {  
 String strMonthAge = **""**;  
 **try**{  
 **if**(!**birthdate**.equals(**""**)){  
 SimpleDateFormat sdf = **new** SimpleDateFormat(**"yyyy-MM-dd"**);  
 strMonthAge = AgeUtils.*getAge*(sdf.parse(**this**.**birthdate**));  
 }  
 }  
 **catch** (Exception ex){  
  
 }  
 **return** strMonthAge;  
 }  
  
 **public int** getMonthAge() **throws** ParseException {  
 **int** monthAge = 0;  
 **try**{  
 **if**(!**birthdate**.equals(**""**)){  
 SimpleDateFormat sdf = **new** SimpleDateFormat(**"yyyy-MM-dd"**);  
 monthAge = AgeUtils.*getMonthAge*(sdf.parse(**this**.**birthdate**));  
 }  
 }  
 **catch** (Exception ex){  
  
 }  
 **return** monthAge;  
 }  
}

**package** com.ChildHealthDiet.app2.context;

**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.utils.SPUtils;  
**public class** UserContext {  
 **final** String **nicknameKey** = **"nicknameKey"**;  
 **final** String **birthdayKey** = **"brithdayKey"**;  
 **private volatile static** UserContext *singleton*;  
 **private** UserContext() {  
 }  
  
 **public static** UserContext getInstance() {  
  
 **if** (*singleton* == **null**) {  
 **synchronized** (UserContext.**class**) {  
 **if** (*singleton* == **null**) {  
 *singleton* = **new** UserContext();  
 }  
 }  
 }  
 **return** *singleton*;  
 }  
  
 **public** Kidinfo getmKidinfo(Context context) {  
 **if**(**mKidinfo** == **null**){  
 **mKidinfo** = **new** Kidinfo(**""**,**""**);  
 }  
 **mKidinfo**.setNickName((String) SPUtils.*get*(context,**nicknameKey**,**""**));  
 **mKidinfo**.setBirthdate((String)SPUtils.*get*(context,**birthdayKey**,**""**));  
 **return mKidinfo**;  
 }  
  
 **public void** setmKidinfo(Kidinfo mKidinfo) {  
 **this**.**mKidinfo** = mKidinfo;  
 }  
  
 **public** Kidinfo getmKidinfo() {  
 **if**(**mKidinfo** == **null**){  
 **mKidinfo** = **new** Kidinfo(**""**,**""**);  
 }  
 **return mKidinfo**;  
 }  
  
 **public void** setmKidinfo(Context context,String strNickName,String strBirthday) {  
 **if**(!strNickName.equals(**""**)){  
 SPUtils.*put*(context,**nicknameKey**,strNickName);  
 }  
 **if**(!strBirthday.equals(**""**)){  
 SPUtils.*put*(context,**birthdayKey**,strBirthday);  
 }  
 }  
  
  
 **private** Kidinfo **mKidinfo** = **null**;  
}

**package** com.ChildHealthDiet.app2.event;  
  
**public class** ChangeTabEvent {  
 **public int tabIndex**;  
  
 **public** ChangeTabEvent(**int** index){  
 **this**.**tabIndex** = index;  
 }  
}

**package** com.ChildHealthDiet.app2.model.bean;  
  
**import** android.os.Parcel;  
**import** android.os.Parcelable;  
**public class** CookRecipeMethod **implements** Parcelable {  
  
 **private** String **img**;  
 **private** String **step**;  
  
 **public** CookRecipeMethod(){  
  
 }  
  
 **public** String getImg() {  
 **return img**;  
 }  
  
 **public void** setImg(String img) {  
 **this**.**img** = img;  
 }  
  
 **public** String getStep() {  
 **return step**;  
 }  
  
 **public void** setStep(String step) {  
 **this**.**step** = step;  
 }  
  
  
 @Override  
 **public int** describeContents() {  
 **return** 0;  
 }  
  
 @Override  
 **public void** writeToParcel(Parcel dest, **int** flags) {  
 dest.writeString(**this**.**img**);  
 dest.writeString(**this**.**step**);  
 }  
  
 **protected** CookRecipeMethod(Parcel in) {  
 **this**.**img** = in.readString();  
 **this**.**step** = in.readString();  
 }  
  
 **public static final** Creator<CookRecipeMethod> ***CREATOR*** = **new** Creator<CookRecipeMethod>() {  
 @Override  
 **public** CookRecipeMethod createFromParcel(Parcel source) {  
 **return new** CookRecipeMethod(source);  
 }  
  
 @Override  
 **public** CookRecipeMethod[] newArray(**int** size) {  
 **return new** CookRecipeMethod[size];  
 }  
 };  
}

**package** com.ChildHealthDiet.app2.model.bean;  
  
**import** android.os.Parcel;  
**import** android.os.Parcelable;  
  
**public class** KeyValueBean **implements** Parcelable {  
  
 String **key**;  
 String **value**;  
 **public** String getKey() {  
 **return key**;  
 }  
 **public void** setKey(String key) {  
 **this**.**key** = key;  
 }  
 **public** String getValue() {  
 **return value**;  
 }  
 **public void** setValue(String value) {  
 **this**.**value** = value;  
 }  
 @Override  
 **public int** describeContents() {  
 **return** 0;  
 }  
 @Override  
 **public void** writeToParcel(Parcel dest, **int** flags) {  
  
 dest.writeString(**this**.**key**);  
 dest.writeString(**this**.**value**);  
 }  
 **protected** KeyValueBean(Parcel in) {  
 **this**.**key** = in.readString();  
 **this**.**value** = in.readString();  
 }  
  
 **public** KeyValueBean(String strKey,String strValue) {  
 **this**.**key** = strKey;  
 **this**.**value** =strValue;  
 }  
  
 **public static final** Parcelable.Creator<KeyValueBean> ***CREATOR*** = **new** Parcelable.Creator<KeyValueBean>() {  
 @Override  
 **public** KeyValueBean createFromParcel(Parcel source) {  
 **return new** KeyValueBean(source);  
 }  
  
 @Override  
 **public** KeyValueBean[] newArray(**int** size) {  
 **return new** KeyValueBean[size];  
 }  
 };  
}

**package** com.ChildHealthDiet.app2.model.bean;  
  
**public class** MineItem {  
 **private** String **key**;  
 **private** String **name**;  
 **private** String **picture**;  
 **private** String **jumpClassName**;  
 **private** String **jumpingParam**;  
  
 **public** String getKey() {  
 **return key**;  
 }  
 **public void** setKey(String key) {  
 **this**.**key** = key;  
 }  
 **public** String getName() {  
 **return name**;  
 }  
 **public void** setName(String name) {  
 **this**.**name** = name;  
 }  
 **public** String getPicture() {  
 **return picture**;  
 }  
 **public void** setPicture(String picture) {  
 **this**.**picture** = picture;  
   
 **public** String getJumpClassName() {  
 **return jumpClassName**;  
 }  
 **public void** setJumpClassName(String jumpClassName) {  
 **this**.**jumpClassName** = jumpClassName;  
 }  
 **public** String getJumpingParam() {  
 **return jumpingParam**;  
 }  
 **public void** setJumpingParam(String jumpingParam) {  
 **this**.**jumpingParam** = jumpingParam;  
 }  
}

**package** com.ChildHealthDiet.app2.model.bean;  
  
**import** android.os.Parcel;  
**import** android.os.Parcelable;  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
**public class** MonthRecipe **implements** Parcelable {  
 **public** String **key**;  
 **public** String **title**;  
 **public** String **subTitle**;  
 **public** List<String> **images**;  
 **public** String getKey() {  
 **return key**;  
 }  
 **public void** setKey(String key) {  
 **this**.**key** = key;  
 }  
 **public** String getTitle() {  
 **return title**;  
 }  
 **public void** setTitle(String title) {  
 **this**.**title** = title;  
 }  
 **public** String getSubTitle() {  
 **return subTitle**;  
 }  
 **public void** setSubTitle(String subTitle) {  
 **this**.**subTitle** = subTitle;  
 }  
 **public** List<String> getImages() {  
 **return images**;  
 }  
 **public void** setImages(List<String> images) {  
 **this**.**images** = images;  
 }  
 @Override  
 **public int** describeContents() {  
 **return** 0;  
 }  
 @Override  
 **public void** writeToParcel(Parcel dest, **int** flags) {  
 dest.writeString(**this**.**key**);  
 dest.writeString(**this**.**title**);  
 dest.writeString(**this**.**subTitle**);  
 dest.writeStringList(**this**.**images**);  
 }  
 **protected** MonthRecipe(Parcel in) {  
 **this**.**key** = in.readString();  
 **this**.**title** = in.readString();  
 **this**.**subTitle** = in.readString();  
 **this**.**images** = **new** ArrayList<String>();  
 in.readStringList(**this**.**images**);  
 }  
   
 **public static final** Parcelable.Creator<MonthRecipe> ***CREATOR*** = **new** Parcelable.Creator<MonthRecipe>() {  
 @Override  
 **public** MonthRecipe createFromParcel(Parcel source) {  
 **return new** MonthRecipe(source);  
 }  
 @Override  
 **public** MonthRecipe[] newArray(**int** size) {  
 **return new** MonthRecipe[size];  
 }  
 };  
}

**package** com.ChildHealthDiet.app2.model.bean;  
  
**import** org.greenrobot.greendao.annotation.Entity;  
**import** org.greenrobot.greendao.annotation.Generated;  
**import** org.greenrobot.greendao.annotation.Id;  
**import** org.greenrobot.greendao.annotation.Property;  
**import** org.greenrobot.greendao.annotation.Transient;  
  
@Entity(  
 nameInDb = **"childrenfood"**,  
 createInDb = **false**)  
**public class** RecipeBean {  
   
 @Property(nameInDb = **"id"**)  
 @Id(autoincrement = **true**)  
 **private** Long **id**;  
 **private** String **name**;  
 **private** String **ingredients**;  
 **private** String **practice**;  
 **private** String **prompt**;  
  
 **private** String **symptoms**;  
 **private** String **month**;  
 **private** String **lock**;  
 **private** String **eattime**;  
 **private** String **type**;  
  
 **private** String **picture**;  
 **private** String **material**;  
 **private** Boolean **collect**;  
 **private** Boolean **basket**;  
 **private** Boolean **bedit**;  
  
 **public** Boolean getShowCheckBox() {  
 **return showCheckBox**;  
 }  
  
 **public void** setShowCheckBox(Boolean showCheckBox) {  
 **this**.**showCheckBox** = showCheckBox;  
 }  
  
 @Transient  
 **public** Boolean **showCheckBox** = **false**;  
  
 **public** Boolean getCanDelete() {  
 **return isCanDelete**;  
 }  
  
 **public void** setCanDelete(Boolean canDelete) {  
 **isCanDelete** = canDelete;  
 }  
  
 @Transient  
 **public** Boolean **isCanDelete** = **false**;  
@Generated(hash = 2062217939)  
**public** RecipeBean(Long id, String name, String ingredients, String practice,  
 String prompt, String symptoms, String month, String lock,  
 String eattime, String type, String picture, String material,  
 Boolean collect, Boolean basket, Boolean bedit) {  
 **this**.**id** = id;  
 **this**.**name** = name;  
 **this**.**ingredients** = ingredients;  
 **this**.**practice** = practice;  
 **this**.**prompt** = prompt;  
 **this**.**symptoms** = symptoms;  
 **this**.**month** = month;  
 **this**.**lock** = lock;  
 **this**.**eattime** = eattime;  
 **this**.**type** = type;  
 **this**.**picture** = picture;  
 **this**.**material** = material;  
 **this**.**collect** = collect;  
 **this**.**basket** = basket;  
 **this**.**bedit** = bedit;  
}  
@Generated(hash = 1492456445)  
**public** RecipeBean() {  
}  
**public** Long getId() {  
 **return this**.**id**;  
}  
**public void** setId(Long id) {  
 **this**.**id** = id;  
}  
**public** String getName() {  
 **return this**.**name**;  
}  
**public void** setName(String name) {  
 **this**.**name** = name;  
}  
**public** String getIngredients() {  
 **return this**.**ingredients**;  
}  
**public void** setIngredients(String ingredients) {  
 **this**.**ingredients** = ingredients;  
}  
**public** String getPractice() {  
 **return this**.**practice**.replaceAll(**"<br>"**,**""**);  
}  
**public void** setPractice(String practice) {  
 **this**.**practice** = practice;  
}  
**public** String getPrompt() {  
 **return this**.**prompt**;  
}  
**public void** setPrompt(String prompt) {  
 **this**.**prompt** = prompt;  
}  
**public** String getSymptoms() {  
 **return this**.**symptoms**;  
}  
**public void** setSymptoms(String symptoms) {  
 **this**.**symptoms** = symptoms;  
}  
**public** String getMonth() {  
 **return this**.**month**;  
}  
**public void** setMonth(String month) {  
 **this**.**month** = month;  
}  
**public** String getLock() {  
 **return this**.**lock**;  
}  
**public void** setLock(String lock) {  
 **this**.**lock** = lock;  
}  
**public** String getEattime() {  
 **return this**.**eattime**;  
}  
**public void** setEattime(String eattime) {  
 **this**.**eattime** = eattime;  
}  
**public** String getType() {  
 **return this**.**type**;  
}  
**public void** setType(String type) {  
 **this**.**type** = type;  
}  
**public** String getPicture() {  
 **return this**.**picture**;  
}  
**public void** setPicture(String picture) {  
 **this**.**picture** = picture;  
}  
**public** String getMaterial() {  
 **return this**.**material**;  
}  
**public void** setMaterial(String material) {  
 **this**.**material** = material;  
}  
**public** Boolean getCollect() {  
 **return this**.**collect**!=**null** && **this**.**collect**;  
}  
**public void** setCollect(Boolean collect) {  
 **this**.**collect** = collect;  
}  
**public** Boolean getBasket() {  
 **return this**.**basket**!=**null** && **this**.**basket**;  
}  
**public void** setBasket(Boolean basket) {  
 **this**.**basket** = basket;  
}  
**public** Boolean getBedit() {  
 **return this**.**bedit**;  
}  
**public void** setBedit(Boolean bedit) {  
 **this**.**bedit** = bedit;  
}

}

**package** com.ChildHealthDiet.app2.model.bean;  
  
**import** java.util.List;  
  
**public class** RecipeCategory {  
 **public** String **name**;  
 **public** List<CategoryName> **categorys**;  
  
 **public class** CategoryName {  
 **public** String **key**;  
 **public** String **name**;  
 }  
}

**package** com.ChildHealthDiet.app2.model.contract;  
  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.model.bean.MonthRecipe;  
**import** java.util.List;  
  
**public interface** MonthRecipeModel {  
  
 List<MonthRecipe> loadMonthRecipe(Context context);  
}

**package** com.ChildHealthDiet.app2.model.contract;  
  
**import** android.content.Context;  
  
**import** com.ChildHealthDiet.app2.model.bean.RecipeBean;  
  
**import** java.util.List;  
  
**public interface** RecipeModel {  
 List<RecipeBean> loadRecipeBeanbyKeywords(Context context,String[] keywords);  
 List<RecipeBean> searchRecipeBeanbyKeyword(Context context,String keyword);  
 List<RecipeBean> loadRecipeBeanbyMonth(Context context,String keyword);  
  
 List<RecipeBean> filterRecipeBeanbyKeyword(Context context,List<Long> ids,String keyword);  
  
 RecipeBean getRecipeById(Context context,**long** id);  
  
 **void** updateRecipe(RecipeBean recipeBean);  
  
 List<RecipeBean> getBasketRecipeBean();  
  
 **void** updateRecipes(RecipeBean[] recipeBeans);  
  
 List<RecipeBean> getColletRecipeBean();  
  
 **void** deleteCollectRecipe(List<RecipeBean> recipeBeans);  
  
 List<RecipeBean> getSymptomsRecipeBean(String strSymptoms);  
  
 List<RecipeBean> getEattimeRecipeBean(String strEatTime,String strMonth);  
  
 List<RecipeBean> getTypeRecipeBean(String strTypeName);  
  
 List<RecipeBean> getIngredientsRecipeBean(String strIngredients);  
  
}

**package** com.ChildHealthDiet.app2.model.contract;  
  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.model.bean.MineItem;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeCategory;  
**import** java.util.List;  
  
**public interface** SimpleModel {  
 String[] loadCategoryField(Context context);  
 List<String> loadFieldByKey(Context context, **int** intKey);  
 List<RecipeCategory> loadCategoryslist(Context context);  
 List<MineItem> loadMineitem(Context context);  
}

**package** com.ChildHealthDiet.app2.model.update;  
  
**import** android.database.Cursor;  
**import** android.text.TextUtils;  
**import** android.util.Log;  
**import** org.greenrobot.greendao.AbstractDao;  
**import** org.greenrobot.greendao.database.Database;  
**import** org.greenrobot.greendao.internal.DaoConfig;  
**import** java.lang.reflect.InvocationTargetException;  
**import** java.lang.reflect.Method;  
**import** java.util.ArrayList;  
**import** java.util.Arrays;  
**import** java.util.List;  
  
**public class** MigrationHelper {  
  
 **private static final** String ***CONVERSION\_CLASS\_NOT\_FOUND\_EXCEPTION*** = **"MIGRATION HELPER - CLASS DOESN'T MATCH WITH THE CURRENT PARAMETERS"**;  
 **private static** MigrationHelper *instance*;  
  
 **public static** MigrationHelper getInstance() {  
 **if** (*instance* == **null**) {  
 *instance* = **new** MigrationHelper();  
 }  
 **return** *instance*;  
 }  
  
 **public void** migrate(Database db, Class<? **extends** AbstractDao<?, ?>>... daoClasses) {  
  
 generateTempTables(db, daoClasses);  
 deleteOriginalTables(db, daoClasses);  
 createOrignalTables(db, daoClasses);  
 restoreData(db, daoClasses);  
 }  
  
 */\*\*  
 \* 生成临时列表  
 \*  
 \** ***@param db*** *\** ***@param daoClasses*** *\*/* **private void** generateTempTables(Database db, Class<? **extends** AbstractDao<?, ?>>... daoClasses) {  
 **for** (**int** i = 0; i < daoClasses.**length**; i++) {  
 DaoConfig daoConfig = **new** DaoConfig(db, daoClasses[i]);  
  
 String divider = **""**;  
 String tableName = daoConfig.**tablename**;  
 String tempTableName = daoConfig.**tablename**.concat(**"\_TEMP"**);  
 ArrayList<String> properties = **new** ArrayList<>();  
  
 StringBuilder createTableStringBuilder = **new** StringBuilder();  
  
 createTableStringBuilder.append(**"CREATE TABLE "**).append(tempTableName).append(**" ("**);  
  
 **for** (**int** j = 0; j < daoConfig.**properties**.**length**; j++) {  
 String columnName = daoConfig.**properties**[j].**columnName**;  
  
 **if** (getColumns(db, tableName).contains(columnName)) {  
 properties.add(columnName);  
  
 String type = **null**;  
  
 **try** {  
 type = getTypeByClass(daoConfig.**properties**[j].**type**);  
 } **catch** (Exception exception) {  
 exception.printStackTrace();  
 }  
  
 createTableStringBuilder.append(divider).append(columnName).append(**" "**).append(type);  
  
 **if** (daoConfig.**properties**[j].**primaryKey**) {  
 createTableStringBuilder.append(**" PRIMARY KEY"**);  
 }  
  
 divider = **","**;  
 }  
 }  
 createTableStringBuilder.append(**");"**);  
  
 db.execSQL(createTableStringBuilder.toString());  
  
 StringBuilder insertTableStringBuilder = **new** StringBuilder();  
  
 insertTableStringBuilder.append(**"INSERT INTO "**).append(tempTableName).append(**" ("**);  
 insertTableStringBuilder.append(TextUtils.*join*(**","**, properties));  
 insertTableStringBuilder.append(**") SELECT "**);  
 insertTableStringBuilder.append(TextUtils.*join*(**","**, properties));  
 insertTableStringBuilder.append(**" FROM "**).append(tableName).append(**";"**);  
  
 db.execSQL(insertTableStringBuilder.toString());  
  
 }  
 }**private void** deleteOriginalTables(Database db, Class<? **extends** AbstractDao<?, ?>>... daoClasses) {  
 **for** (Class<? **extends** AbstractDao<?, ?>> daoClass : daoClasses) {  
 **try** {  
 Method method = daoClass.getMethod(**"dropTable"**, Database.**class**, **boolean**.**class**);  
 method.invoke(**null**, db, **true**);  
 } **catch** (IllegalAccessException e) {  
 e.printStackTrace();  
 } **catch** (InvocationTargetException e) {  
 e.printStackTrace();  
 } **catch** (NoSuchMethodException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
**private void** createOrignalTables(Database db, Class<? **extends** AbstractDao<?, ?>>... daoClasses) {  
 **for** (Class<? **extends** AbstractDao<?, ?>> daoClass : daoClasses) {  
 **try** {  
 Method method = daoClass.getMethod(**"createTable"**, Database.**class**, **boolean**.**class**);  
 method.invoke(**null**, db, **false**);  
 } **catch** (IllegalAccessException e) {  
 e.printStackTrace();  
 } **catch** (InvocationTargetException e) {  
 e.printStackTrace();  
 } **catch** (NoSuchMethodException e) {  
 e.printStackTrace();  
 }  
 }  
 }**private void** restoreData(Database db, Class<? **extends** AbstractDao<?, ?>>... daoClasses) {  
 **for** (**int** i = 0; i < daoClasses.**length**; i++) {  
 DaoConfig daoConfig = **new** DaoConfig(db, daoClasses[i]);  
 String tableName = daoConfig.**tablename**;  
 String tempTableName = daoConfig.**tablename**.concat(**"\_TEMP"**);  
 ArrayList<String> properties = **new** ArrayList();  
  
 **for** (**int** j = 0; j < daoConfig.**properties**.**length**; j++) {  
 String columnName = daoConfig.**properties**[j].**columnName**;  
  
 **if** (getColumns(db, tempTableName).contains(columnName)) {  
 properties.add(columnName);  
 }  
 }  
  
 StringBuilder insertTableStringBuilder = **new** StringBuilder();  
  
 insertTableStringBuilder.append(**"INSERT INTO "**).append(tableName).append(**" ("**);  
 insertTableStringBuilder.append(TextUtils.*join*(**","**, properties));  
 insertTableStringBuilder.append(**") SELECT "**);  
 insertTableStringBuilder.append(TextUtils.*join*(**","**, properties));  
 insertTableStringBuilder.append(**" FROM "**).append(tempTableName).append(**";"**);  
  
 StringBuilder dropTableStringBuilder = **new** StringBuilder();  
 dropTableStringBuilder.append(**"DROP TABLE "**).append(tempTableName);  
 db.execSQL(insertTableStringBuilder.toString());  
 db.execSQL(dropTableStringBuilder.toString());  
 }  
 }  
 **private** String getTypeByClass(Class<?> type) **throws** Exception {  
 **if** (type.equals(String.**class**)) {  
 **return "TEXT"**;  
 }  
 **if** (type.equals(Long.**class**) || type.equals(Integer.**class**) || type.equals(**long**.**class**)) {  
 **return "INTEGER"**;  
 }  
 **if** (type.equals(Boolean.**class**)) {  
 **return "BOOLEAN"**;  
 }  
  
 Exception exception = **new** Exception(***CONVERSION\_CLASS\_NOT\_FOUND\_EXCEPTION***.concat(**" - Class: "**).concat(type.toString()));  
 exception.printStackTrace();  
 **throw** exception;  
 }  
 **private** List<String> getColumns(Database db, String tableName) {  
 List<String> columns = **new** ArrayList<>();  
 Cursor cursor = **null**;  
 **try** {  
 cursor = db.rawQuery(**"SELECT \* FROM "** + tableName + **" limit 1"**, **null**);  
 **if** (cursor != **null**) {  
 columns = **new** ArrayList<>(Arrays.*asList*(cursor.getColumnNames()));  
 }  
 } **catch** (Exception e) {  
 Log.*v*(tableName, e.getMessage(), e);  
 e.printStackTrace();  
 } **finally** {  
 **if** (cursor != **null**)  
 cursor.close();  
 }  
 **return** columns;  
 }  
}

**package** com.ChildHealthDiet.app2.model.update;  
  
**import** android.database.Cursor;  
**import** android.text.TextUtils;  
**import** android.util.Log;  
**import** org.greenrobot.greendao.AbstractDao;  
**import** org.greenrobot.greendao.database.Database;  
**import** org.greenrobot.greendao.internal.DaoConfig;  
**import** java.lang.reflect.InvocationTargetException;  
**import** java.lang.reflect.Method;  
**import** java.util.ArrayList;  
**import** java.util.Arrays;  
**import** java.util.List;  
  
**public class** Update2Helper {  
 **private static final** String ***TAG*** = **"BookChapterHelper"**;  
 **private static final** String ***CONVERSION\_CLASS\_NOT\_FOUND\_EXCEPTION*** = **"MIGRATION HELPER - CLASS DOESN'T MATCH WITH THE CURRENT PARAMETERS"**;  
  
 **private static final** String ***DIVIDER*** = **","**;  
 **private static final** String ***QUOTE*** = **"'%s'"**;  
  
 **private static** Update2Helper *instance*;  
  
 **public static** Update2Helper getInstance() {  
 **if** (*instance* == **null**) {  
 *instance* = **new** Update2Helper();  
 }  
 **return** *instance*;  
 }  
 **public void** update(Database db) {  
 updateCollBook(db);  
 updateBookChapter(db);  
 }  
 **private void** updateBookChapter(Database db) {  
 }  
 **private void** updateCollBook(Database db) {  
 }**private void** generateTempTables(Database db, Class<? **extends** AbstractDao<?, ?>> bookChapterClass) {  
  
 }  
**private void** deleteOriginalTables(Database db, Class<? **extends** AbstractDao<?, ?>> bookChapterClass) {  
 **try** {  
 Method method = bookChapterClass.getMethod(**"dropTable"**, Database.**class**, **boolean**.**class**);  
 method.invoke(**null**, db, **true**);  
 } **catch** (IllegalAccessException e) {  
 e.printStackTrace();  
 } **catch** (InvocationTargetException e) {  
 e.printStackTrace();  
 } **catch** (NoSuchMethodException e) {  
 e.printStackTrace();  
 }  
 }  
**private void** createOrignalTables(Database db, Class<? **extends** AbstractDao<?, ?>> bookChapterClass) {  
 **try** {  
 Method method = bookChapterClass.getMethod(**"createTable"**, Database.**class**, **boolean**.**class**);  
 method.invoke(**null**, db, **false**);  
 } **catch** (IllegalAccessException e) {  
 e.printStackTrace();  
 } **catch** (InvocationTargetException e) {  
 e.printStackTrace();  
 } **catch** (NoSuchMethodException e) {  
 e.printStackTrace();  
 }  
 }**private void** restoreData(Database db, Class<? **extends** AbstractDao<?, ?>> bookChapterClass) {  
 DaoConfig daoConfig = **new** DaoConfig(db, bookChapterClass);  
 String tableName = daoConfig.**tablename**;  
 String tempTableName = daoConfig.**tablename**.concat(**"\_TEMP"**);  
 ArrayList<String> properties = **new** ArrayList();  
  
 **for** (**int** j = 0; j < daoConfig.**properties**.**length**; j++) {  
 String columnName = daoConfig.**properties**[j].**columnName**;  
 **if** (getColumns(db, tableName).contains(columnName)) {  
 properties.add(columnName);  
 }  
 }  
 StringBuilder insertTableStringBuilder = **new** StringBuilder();  
 insertTableStringBuilder.append(**"INSERT INTO "**).append(tableName).append(**" ("**);  
 insertTableStringBuilder.append(TextUtils.*join*(**","**, properties));  
 insertTableStringBuilder.append(**") SELECT "**);  
 insertTableStringBuilder.append(TextUtils.*join*(**","**, properties));  
 insertTableStringBuilder.append(**" FROM "**).append(tempTableName).append(**";"**);  
  
 Log.*d*(***TAG***, **"restoreData: "** + insertTableStringBuilder.toString());  
  
 StringBuilder dropTableStringBuilder = **new** StringBuilder();  
 dropTableStringBuilder.append(**"DROP TABLE "**).append(tempTableName);  
 db.execSQL(insertTableStringBuilder.toString());  
 db.execSQL(dropTableStringBuilder.toString());  
 }  
 **private** String getTypeByClass(Class<?> type) **throws** Exception {  
 **if** (type.equals(String.**class**)) {  
 **return "TEXT"**;  
 }  
 **if** (type.equals(Long.**class**) || type.equals(Integer.**class**) || type.equals(**long**.**class**)) {  
 **return "INTEGER"**;  
 }  
 **if** (type.equals(**boolean**.**class**)) {  
 **return "BOOLEAN"**;  
 }  
 Exception exception = **new** Exception(***CONVERSION\_CLASS\_NOT\_FOUND\_EXCEPTION***.concat(**" - Class: "**).concat(type.toString()));  
 exception.printStackTrace();  
 **throw** exception;  
 }  
 **private** List<String> getColumns(Database db, String tableName) {  
 List<String> columns = **new** ArrayList<>();  
 Cursor cursor = **null**;  
 **try** {  
 cursor = db.rawQuery(**"SELECT \* FROM "** + tableName + **" limit 1"**, **null**);  
 **if** (cursor != **null**) {  
 columns = **new** ArrayList<>(Arrays.*asList*(cursor.getColumnNames()));  
 }  
 } **catch** (Exception e) {  
 Log.*v*(tableName, e.getMessage(), e);  
 e.printStackTrace();  
 } **finally** {  
 **if** (cursor != **null**)  
 cursor.close();  
 }  
 **return** columns;  
 }  
}

**package** com.ChildHealthDiet.app2.model;  
  
**import** com.ChildHealthDiet.app2.RecipeApplication;  
**import** com.ChildHealthDiet.app2.model.bean.DaoMaster;  
**import** com.ChildHealthDiet.app2.model.bean.DaoSession;  
**import** org.greenrobot.greendao.database.Database;  
  
**public class** DaoDbHelper {  
 **private static final** String ***DB\_NAME*** = **"babydietfood.db"**;  
  
 **private static volatile** DaoDbHelper *sInstance*;  
 **private** Database **mDb**;  
 **private** DaoMaster **mDaoMaster**;  
 **private** DaoSession **mDaoSession**;  
 **private** DaoDbHelper(){  
 *//封装数据库的创建、更新、删除* DaoMaster.DevOpenHelper helper = **new** DaoMaster.DevOpenHelper(RecipeApplication.*getContext*(), ***DB\_NAME***);  
 Database mDb = helper.getEncryptedWritableDb(**"key123"**);  
 **mDaoSession** = **new** DaoMaster(mDb).newSession();  
 }  
  
 **public static** DaoDbHelper getInstance(){  
 **if** (*sInstance* == **null**){  
 **synchronized** (DaoDbHelper.**class**){  
 **if** (*sInstance* == **null**){  
 *sInstance* = **new** DaoDbHelper();  
 }  
 }  
 }  
 **return** *sInstance*;  
 }  
 **public** DaoSession getSession(){  
 **return mDaoSession**;  
   
 **public** Database getDatabase(){  
 **return mDb**;  
 }  
}

**package** com.ChildHealthDiet.app2.model;  
  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.R;  
**import** com.ChildHealthDiet.app2.model.bean.MonthRecipe;  
**import** com.ChildHealthDiet.app2.model.contract.MonthRecipeModel;  
**import** com.ChildHealthDiet.app2.utils.FileUtils;  
**import** com.google.gson.Gson;  
**import** com.google.gson.reflect.TypeToken;  
**import** java.util.List;  
  
**public class** MonthRecipeModelImpl **implements** MonthRecipeModel {  
  
 @Override  
 **public** List<MonthRecipe> loadMonthRecipe(Context context) {  
 String jsonFilePath = context.getString(R.string.***month\_json\_path***);  
 String monthJson = FileUtils.*getJson*(context,jsonFilePath);  
 Gson gson = **new** Gson();  
 List<MonthRecipe> monthRecipeList = (List<MonthRecipe>)gson .fromJson(monthJson,  
 **new** TypeToken<List<MonthRecipe>>(){}.getType());  
 **return** monthRecipeList;  
 }  
  
}

**package** com.ChildHealthDiet.app2.model;  
  
**import** android.content.Context;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** com.ChildHealthDiet.app2.model.bean.DaoMaster;  
**import** com.ChildHealthDiet.app2.model.update.Update2Helper;  
**import** org.greenrobot.greendao.database.Database;  
  
  
**public class** MyOpenHelper **extends** DaoMaster.DevOpenHelper{  
 **public** MyOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory) {  
 **super**(context, name, factory);  
 }  
  
 @Override  
 **public void** onUpgrade(Database db, **int** oldVersion, **int** newVersion) {  
 *// 跨版本更新策略* **switch** (oldVersion){  
 **case** 1:  
 *// 暂无 1.0* **case** 2:  
 *// 更新数据到 3.0* Update2Helper.*getInstance*().update(db);  
 **default**:  
 **break**;  
 }  
 }  
}

**package** com.ChildHealthDiet.app2.model;  
  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeBean;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeBeanDao;  
**import** com.ChildHealthDiet.app2.model.contract.RecipeModel;  
**import** org.greenrobot.greendao.query.QueryBuilder;  
**import** java.util.List;  
  
**public class** RecipeModelImpl **implements** RecipeModel {  
  
 @Override  
 **public** List<RecipeBean> loadRecipeBeanbyKeywords(Context context, String[] keywords){  
 **return null**;  
 }  
 @Override  
 **public** List<RecipeBean> searchRecipeBeanbyKeyword(Context context,String keyword){  
 RecipeBeanDao recipeBeanDao = DaoDbHelper.*getInstance*().getSession().getRecipeBeanDao();  
 **return** recipeBeanDao.queryBuilder().whereOr(RecipeBeanDao.Properties.***Name***.like(**"%"**+keyword+**"%"**),  
 RecipeBeanDao.Properties.***Ingredients***.like(**"%"**+keyword+**"%"**),  
 RecipeBeanDao.Properties.***Eattime***.like(**"%"**+keyword+**"%"**),  
 RecipeBeanDao.Properties.***Prompt***.like(**"%"**+keyword+**"%"**),  
 RecipeBeanDao.Properties.***Type***.like(**"%"**+keyword+**"%"**),  
 RecipeBeanDao.Properties.***Practice***.like(**"%"**+keyword+**"%"**),  
 RecipeBeanDao.Properties.***Material***.like(**"%"**+keyword+**"%"**),  
 RecipeBeanDao.Properties.***Symptoms***.like(**"%"**+keyword+**"%"**))  
 .orderAsc(RecipeBeanDao.Properties.***Month***).list();  
 }  
 @Override  
 **public** List<RecipeBean> filterRecipeBeanbyKeyword(Context context,List<Long> ids,String keyword){  
 RecipeBeanDao recipeBeanDao = DaoDbHelper.*getInstance*().getSession().getRecipeBeanDao();  
 QueryBuilder qb = recipeBeanDao.queryBuilder();  
 **return** qb.where(  
 qb.and(RecipeBeanDao.Properties.***Id***.in(ids),RecipeBeanDao.Properties.***Id***.isNotNull()),  
 qb.or(RecipeBeanDao.Properties.***Name***.like(**"%"**+keyword+**"%"**),  
 RecipeBeanDao.Properties.***Ingredients***.like(**"%"**+keyword+**"%"**),  
 RecipeBeanDao.Properties.***Eattime***.like(**"%"**+keyword+**"%"**),  
 RecipeBeanDao.Properties.***Prompt***.like(**"%"**+keyword+**"%"**),  
 RecipeBeanDao.Properties.***Type***.like(**"%"**+keyword+**"%"**),  
 RecipeBeanDao.Properties.***Practice***.like(**"%"**+keyword+**"%"**),  
 RecipeBeanDao.Properties.***Material***.like(**"%"**+keyword+**"%"**),  
 RecipeBeanDao.Properties.***Symptoms***.like(**"%"**+keyword+**"%"**))  
 ).orderAsc(RecipeBeanDao.Properties.***Month***).list();  
 }  
@Override  
 **public** List<RecipeBean> loadRecipeBeanbyMonth(Context context,String keyword){  
 RecipeBeanDao recipeBeanDao = DaoDbHelper.*getInstance*().getSession().getRecipeBeanDao();  
 **return** recipeBeanDao.queryBuilder().where(RecipeBeanDao.Properties.***Month***.eq(keyword)).list();  
 }  
  
 @Override  
 **public** RecipeBean getRecipeById(Context context,**long** id){  
 RecipeBeanDao recipeBeanDao = DaoDbHelper.*getInstance*().getSession().getRecipeBeanDao();

**return** recipeBeanDao.load(id);  
  
 }  
 @Override  
 **public void** updateRecipe(RecipeBean recipeBean){  
 RecipeBeanDao recipeBeanDao = DaoDbHelper.*getInstance*().getSession().getRecipeBeanDao();  
 recipeBeanDao.update(recipeBean);  
 }  
 @Override  
 **public void** updateRecipes(RecipeBean[] recipeBeans){  
 RecipeBeanDao recipeBeanDao = DaoDbHelper.*getInstance*().getSession().getRecipeBeanDao();  
 recipeBeanDao.updateInTx(recipeBeans);  
 }  
 @Override  
 **public** List<RecipeBean> getBasketRecipeBean(){  
 RecipeBeanDao recipeBeanDao = DaoDbHelper.*getInstance*().getSession().getRecipeBeanDao();  
 QueryBuilder qb = recipeBeanDao.queryBuilder();  
 **return** qb.where(RecipeBeanDao.Properties.***Basket***.eq(**true**)).orderAsc(RecipeBeanDao.Properties.***Month***).list();  
 }  
 @Override  
 **public** List<RecipeBean> getColletRecipeBean(){  
 RecipeBeanDao recipeBeanDao = DaoDbHelper.*getInstance*().getSession().getRecipeBeanDao();  
 QueryBuilder qb = recipeBeanDao.queryBuilder();  
 **return** qb.where(RecipeBeanDao.Properties.***Collect***.eq(**true**)).orderAsc(RecipeBeanDao.Properties.***Month***).list();  
 }  
 @Override  
 **public void** deleteCollectRecipe(List<RecipeBean> recipeBeans) {  
 **if**(recipeBeans != **null**){  
 RecipeBeanDao recipeBeanDao = DaoDbHelper.*getInstance*().getSession().getRecipeBeanDao();  
 **for**(RecipeBean itemRecipe : recipeBeans) {**if**(itemRecipe.**isCanDelete**){  
 itemRecipe.setCollect(**false**);  
 itemRecipe.setShowCheckBox(**false**);  
 recipeBeanDao.update(itemRecipe);  
 }  
 }  
 }  
 }  
 @Override  
 **public** List<RecipeBean> getSymptomsRecipeBean(String strSymptoms){  
 RecipeBeanDao recipeBeanDao = DaoDbHelper.*getInstance*().getSession().getRecipeBeanDao();  
 QueryBuilder qb = recipeBeanDao.queryBuilder();  
 **return** qb.where(RecipeBeanDao.Properties.***Symptoms***.like(**"%"**+strSymptoms+**"%"**)).orderAsc(RecipeBeanDao.Properties.***Month***).list();  
 }  
  
 @Override  
 **public** List<RecipeBean> getEattimeRecipeBean(String strEatTime,String strMonth){  
 RecipeBeanDao recipeBeanDao = DaoDbHelper.*getInstance*().getSession().getRecipeBeanDao();  
 QueryBuilder qb = recipeBeanDao.queryBuilder();  
 **if**(strMonth.equals(**""**)){  
 **return** qb.where(RecipeBeanDao.Properties.***Eattime***.like(**"%"**+strEatTime+**"%"**))  
 .orderAsc(RecipeBeanDao.Properties.***Month***).list();  
 }  
 **else** {  
 **return** qb.where(  
 qb.and(RecipeBeanDao.Properties.***Month***.eq(strMonth),  
 RecipeBeanDao.Properties.***Eattime***.like(**"%"**+strEatTime+**"%"**)))  
 .orderAsc(RecipeBeanDao.Properties.***Month***).list();  
   
 }  
 @Override  
 **public** List<RecipeBean> getTypeRecipeBean(String strTypeName){  
 RecipeBeanDao recipeBeanDao = DaoDbHelper.*getInstance*().getSession().getRecipeBeanDao();  
 QueryBuilder qb = recipeBeanDao.queryBuilder();  
 **return** qb.where(RecipeBeanDao.Properties.***Type***.like(**"%"**+strTypeName+**"%"**)).orderAsc(RecipeBeanDao.Properties.***Month***).list();  
 }  
 @Override  
 **public** List<RecipeBean> getIngredientsRecipeBean(String strIngredients){  
 RecipeBeanDao recipeBeanDao = DaoDbHelper.*getInstance*().getSession().getRecipeBeanDao();  
 QueryBuilder qb = recipeBeanDao.queryBuilder();  
 **return** qb.where(RecipeBeanDao.Properties.***Ingredients***.like(**"%"**+strIngredients+**"%"**))  
 .orderAsc(RecipeBeanDao.Properties.***Month***).list();  
 }  
}

**package** com.ChildHealthDiet.app2.model;  
  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.R;  
**import** com.ChildHealthDiet.app2.model.bean.MineItem;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeCategory;  
**import** com.ChildHealthDiet.app2.model.contract.SimpleModel;  
**import** com.ChildHealthDiet.app2.utils.FileUtils;  
**import** com.google.gson.Gson;  
**import** com.google.gson.reflect.TypeToken;  
**import** java.util.Arrays;  
**import** java.util.List;  
  
**public class** SimpleModelImpl **implements** SimpleModel {  
  
 @Override  
 **public** String[] loadCategoryField(Context context) {  
 String[] gvCategorys = context.getResources().getStringArray(R.array.***gv\_category***);  
 **return** gvCategorys;  
 }  
 @Override  
 **public** List<String> loadFieldByKey(Context context, **int** intKey){  
 String[] strs = context.getResources().getStringArray(intKey);  
 **return** Arrays.*asList*(strs);  
 }  
  
 @Override  
 **public** List<RecipeCategory> loadCategoryslist(Context context){  
 String jsonFilePath = context.getString(R.string.***category\_json\_path***);  
 String monthJson = FileUtils.*getJson*(context,jsonFilePath);  
 Gson gson = **new** Gson();  
 List<RecipeCategory> categoryList = (List<RecipeCategory>)gson.fromJson(monthJson,  
 **new** TypeToken<List<RecipeCategory>>(){}.getType());  
 **return** categoryList;  
 }  
  
 @Override  
 **public** List<MineItem> loadMineitem(Context context){  
 String jsonFilePath = context.getString(R.string.***mineitem\_json\_path***);  
 String monthJson = FileUtils.*getJson*(context,jsonFilePath);  
 Gson gson = **new** Gson();  
 List<MineItem> mineItemList = (List<MineItem>)gson.fromJson(monthJson,  
 **new** TypeToken<List<MineItem>>(){}.getType());  
 **return** mineItemList;  
 }  
}

**package** com.ChildHealthDiet.app2.presenter.contract;  
  
**import** com.ChildHealthDiet.app2.model.bean.RecipeBean;  
**import** com.ChildHealthDiet.app2.ui.base.BaseContract;  
**import** java.util.List;  
  
**public interface** BasketContract {  
  
 **interface** View **extends** BaseContract.BaseView{  
 **void** updateBasketRecipeList(List<RecipeBean> recipeBeans);  
 }  
  
 **interface** Presenter **extends** BaseContract.BasePresenter<View>{  
 **void** getBasketRecipeList();  
 **void** updateRecipes(RecipeBean[] aryRecipeBean);  
 }  
}

**package** com.ChildHealthDiet.app2.presenter.contract;  
  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeCategory;  
**import** com.ChildHealthDiet.app2.ui.base.BaseContract;  
**import** java.util.List;  
**public interface** CategorysContract {  
  
 **interface** View **extends** BaseContract.BaseView{  
 **void** updateCategorysList(List<RecipeCategory> categoryslist);  
 }  
  
 **interface** Presenter **extends** BaseContract.BasePresenter<View>{  
 **void** getCategorysList(Context context);  
 }  
}

**package** com.ChildHealthDiet.app2.presenter.contract;  
  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.model.bean.MonthRecipe;  
**import** com.ChildHealthDiet.app2.ui.base.BaseContract;  
**import** java.util.List;  
  
**public interface** FragmentHomeContract {  
  
 **interface** View **extends** BaseContract.BaseView{  
 **void** updateCategory(String[] aryCategory);  
 **void** updateMonthRecipeMenu(List<MonthRecipe> monthRecipes);  
 }  
 **interface** Presenter **extends** BaseContract.BasePresenter<View>{  
 **void** loadMonthRecipe(Context context);  
 **void** loadCategoryField(Context context);  
 }  
}

**package** com.ChildHealthDiet.app2.presenter.contract;  
  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.model.bean.MineItem;  
**import** com.ChildHealthDiet.app2.ui.base.BaseContract;  
**import** java.util.List;  
**public interface** MineContract {  
  
 **interface** View **extends** BaseContract.BaseView{  
 **void** updateMineItem(List<MineItem> mineItems);  
 }  
  
 **interface** Presenter **extends** BaseContract.BasePresenter<View>{  
 **void** getMineItems(Context context);  
 }  
}

**package** com.ChildHealthDiet.app2.presenter.contract;  
  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeBean;  
**import** com.ChildHealthDiet.app2.ui.base.BaseContract;  
**public interface** RecipeDetailContract {  
  
 **interface** View **extends** BaseContract.BaseView{  
 **void** updateRecipe(RecipeBean recipeBean);  
 }  
 **interface** Presenter **extends** BaseContract.BasePresenter<View>{  
 **void** getRecipeById(Context context,**long** id);  
 **void** updateRecipeBeanData(Context context,RecipeBean recipeBean);  
 }  
}

**package** com.ChildHealthDiet.app2.presenter.contract;  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeBean;  
**import** com.ChildHealthDiet.app2.ui.base.BaseContract;  
**import** java.util.List;  
**public interface** RecipeKeywordListContract {  
  
 **interface** View **extends** BaseContract.BaseView{  
 **void** updateRecipe(List<RecipeBean> recipeBeans);  
 }  
  
 **interface** Presenter **extends** BaseContract.BasePresenter<View>{  
 **void** loadRecipeBeanbyMonth(Context context,String strKey);  
 **void** loadRecipeByKeyword(Context context,String strKey);  
 **void** filterByKeyword(Context context,List<Long> ids,String strKey);  
 **void** loadCollectRecipe();  
 **void** deleteCollectRecipe(List<RecipeBean> recipeBeans);  
 **void** loadSymptomsRecipeBean(String strSymptoms);  
 **void** loadEattimeRecipeBean(Context context,String strEatTime);  
 **void** loadTypeRecipeBean(String strType);  
 **void** loadIngredientsRecipeBean(String strIngredients);  
 }  
}

**package** com.ChildHealthDiet.app2.presenter;  
  
**import** com.ChildHealthDiet.app2.model.RecipeModelImpl;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeBean;  
**import** com.ChildHealthDiet.app2.model.contract.RecipeModel;  
**import** com.ChildHealthDiet.app2.presenter.contract.BasketContract;  
**import** com.ChildHealthDiet.app2.ui.base.RxPresenter;

**import** java.util.List;  
**import** io.reactivex.Single;  
**import** io.reactivex.SingleEmitter;  
**import** io.reactivex.SingleOnSubscribe;  
**import** io.reactivex.android.schedulers.AndroidSchedulers;  
**import** io.reactivex.disposables.Disposable;  
**import** io.reactivex.functions.Consumer;  
**import** io.reactivex.schedulers.Schedulers;  
  
**public class** BasketPresenter **extends** RxPresenter<BasketContract.View>  
 **implements** BasketContract.Presenter {  
   
 @Override  
 **public void** getBasketRecipeList() {  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<List<RecipeBean>>() {  
 @Override  
 **public void** subscribe(SingleEmitter<List<RecipeBean>> emitter) **throws** Exception {  
 RecipeModel recipeModel = **new** RecipeModelImpl();  
 emitter.onSuccess(recipeModel.getBasketRecipeBean());  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<List<RecipeBean>>() {  
 @Override  
 **public void** accept(List<RecipeBean> recipeBeans) **throws** Exception {  
 BasketPresenter.**this**.getView().updateBasketRecipeList(recipeBeans);  
 }  
 });  
 addDisposable(disposable);  
 }  
  
 @Override  
 **public void** updateRecipes(RecipeBean[] aryRecipeBean){  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<List<RecipeBean>>() {  
 @Override  
 **public void** subscribe(SingleEmitter<List<RecipeBean>> emitter) **throws** Exception {  
 RecipeModel recipeModel = **new** RecipeModelImpl();  
 recipeModel.updateRecipes(aryRecipeBean);  
 emitter.onSuccess(recipeModel.getBasketRecipeBean());  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<List<RecipeBean>>() {  
 @Override  
 **public void** accept(List<RecipeBean> recipeBeans) **throws** Exception {  
 BasketPresenter.**this**.getView().updateBasketRecipeList(recipeBeans);  
 }  
 });  
 addDisposable(disposable);  
 }  
}

**package** com.ChildHealthDiet.app2.presenter;  
  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.model.SimpleModelImpl;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeCategory;  
**import** com.ChildHealthDiet.app2.model.contract.SimpleModel;  
**import** com.ChildHealthDiet.app2.presenter.contract.CategorysContract;  
**import** com.ChildHealthDiet.app2.ui.base.RxPresenter;  
**import** java.util.List;  
**import** io.reactivex.Single;  
**import** io.reactivex.SingleEmitter;  
**import** io.reactivex.SingleOnSubscribe;  
**import** io.reactivex.android.schedulers.AndroidSchedulers;  
**import** io.reactivex.disposables.Disposable;  
**import** io.reactivex.functions.Consumer;  
**import** io.reactivex.schedulers.Schedulers;  
  
**public class** FragmentCategoryPresenter **extends** RxPresenter<CategorysContract.View>  
 **implements** CategorysContract.Presenter {  
  
 @Override  
 **public void** getCategorysList(Context context) {  
  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<List<RecipeCategory>>() {  
 @Override  
 **public void** subscribe(SingleEmitter<List<RecipeCategory>> emitter) **throws** Exception {  
 SimpleModel simpleModel = **new** SimpleModelImpl();  
 emitter.onSuccess(simpleModel.loadCategoryslist(context));  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<List<RecipeCategory>>() {  
 @Override  
 **public void** accept(List<RecipeCategory> allcategoryslist) **throws** Exception {  
 FragmentCategoryPresenter.**this**.getView().updateCategorysList(allcategoryslist);  
 }  
 });  
 addDisposable(disposable);  
  
 }  
}

**package** com.ChildHealthDiet.app2.presenter;  
  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.model.MonthRecipeModelImpl;  
**import** com.ChildHealthDiet.app2.model.SimpleModelImpl;  
**import** com.ChildHealthDiet.app2.model.bean.MonthRecipe;  
**import** com.ChildHealthDiet.app2.model.contract.MonthRecipeModel;  
**import** com.ChildHealthDiet.app2.model.contract.SimpleModel;  
**import** com.ChildHealthDiet.app2.presenter.contract.FragmentHomeContract;  
**import** com.ChildHealthDiet.app2.ui.base.RxPresenter;  
**import** java.util.List;  
**import** io.reactivex.Single;  
**import** io.reactivex.SingleEmitter;  
**import** io.reactivex.SingleOnSubscribe;  
**import** io.reactivex.android.schedulers.AndroidSchedulers;  
**import** io.reactivex.disposables.Disposable;  
**import** io.reactivex.functions.Consumer;  
**import** io.reactivex.schedulers.Schedulers;  
  
**public class** FragmentHomePresenter **extends** RxPresenter<FragmentHomeContract.View>  
 **implements** FragmentHomeContract.Presenter {  
  
 @Override  
 **public void** loadMonthRecipe(Context context) {  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<List<MonthRecipe>>() {  
 @Override  
 **public void** subscribe(SingleEmitter<List<MonthRecipe>> emitter) **throws** Exception {  
MonthRecipeModel monthRecipeModelImpl = **new** MonthRecipeModelImpl();  
 emitter.onSuccess(monthRecipeModelImpl.loadMonthRecipe(context));  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<List<MonthRecipe>>() {  
 @Override  
 **public void** accept(List<MonthRecipe> listMonthRecipe) **throws** Exception {  
 FragmentHomePresenter.**this**.getView().updateMonthRecipeMenu(listMonthRecipe);  
 }  
 });  
 addDisposable(disposable);  
 }  
  
 @Override  
 **public void** loadCategoryField(Context context) {  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<String[]>() {  
 @Override  
 **public void** subscribe(SingleEmitter<String[]> emitter) **throws** Exception {  
 SimpleModel simpleModel = **new** SimpleModelImpl();  
 emitter.onSuccess(simpleModel.loadCategoryField(context));  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<String[]>() {  
 @Override  
 **public void** accept(String[] strCategorys) **throws** Exception {  
 FragmentHomePresenter.**this**.getView().updateCategory(strCategorys);  
 }  
 });  
 addDisposable(disposable);  
 }  
}

**package** com.ChildHealthDiet.app2.presenter;  
  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.model.SimpleModelImpl;  
**import** com.ChildHealthDiet.app2.model.bean.MineItem;  
**import** com.ChildHealthDiet.app2.model.contract.SimpleModel;  
**import** com.ChildHealthDiet.app2.presenter.contract.MineContract;  
**import** com.ChildHealthDiet.app2.ui.base.RxPresenter;  
**import** java.util.List;  
**import** io.reactivex.Single;  
**import** io.reactivex.SingleEmitter;  
**import** io.reactivex.SingleOnSubscribe;  
**import** io.reactivex.android.schedulers.AndroidSchedulers;  
**import** io.reactivex.disposables.Disposable;  
**import** io.reactivex.functions.Consumer;  
**import** io.reactivex.schedulers.Schedulers;  
  
**public class** MinePresenter **extends** RxPresenter<MineContract.View>  
 **implements** MineContract.Presenter {  
  
 @Override  
 **public void** getMineItems(Context context) {  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<List<MineItem>>() {  
 @Override  
 **public void** subscribe(SingleEmitter<List<MineItem>> emitter) **throws** Exception {  
 SimpleModel simpleModel = **new** SimpleModelImpl();  
 emitter.onSuccess(simpleModel.loadMineitem(context));  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<List<MineItem>>() {  
 @Override  
 **public void** accept(List<MineItem> mineItems) **throws** Exception {  
 MinePresenter.**this**.getView().updateMineItem(mineItems);  
 }  
 });  
 addDisposable(disposable);  
 }  
}

**package** com.ChildHealthDiet.app2.presenter;  
  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.model.RecipeModelImpl;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeBean;  
**import** com.ChildHealthDiet.app2.model.contract.RecipeModel;  
**import** com.ChildHealthDiet.app2.presenter.contract.RecipeDetailContract;  
**import** com.ChildHealthDiet.app2.ui.base.RxPresenter;  
**import** com.ChildHealthDiet.app2.ui.categorys.RECIPETYPE;  
**import** io.reactivex.Single;  
**import** io.reactivex.SingleEmitter;  
**import** io.reactivex.SingleOnSubscribe;  
**import** io.reactivex.android.schedulers.AndroidSchedulers;  
**import** io.reactivex.disposables.Disposable;  
**import** io.reactivex.functions.Consumer;  
**import** io.reactivex.schedulers.Schedulers;  
  
**public class** RecipeDetailPresenter **extends** RxPresenter<RecipeDetailContract.View>  
 **implements** RecipeDetailContract.Presenter {  
  
 RECIPETYPE **recipeType** = RECIPETYPE.***None***;  
 @Override  
 **public void** getRecipeById(Context context,**long** id){  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<RecipeBean>() {  
 @Override  
 **public void** subscribe(SingleEmitter<RecipeBean> emitter) **throws** Exception {  
 RecipeModel recipeModelImpl = **new** RecipeModelImpl();  
 emitter.onSuccess(recipeModelImpl.getRecipeById(context,id));  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<RecipeBean>() {  
 @Override  
 **public void** accept(RecipeBean recipeBean) **throws** Exception {  
 RecipeDetailPresenter.**this**.getView().updateRecipe(recipeBean);  
 }  
 });  
 addDisposable(disposable);  
 }  
 @Override  
 **public void** updateRecipeBeanData(Context context,RecipeBean recipeBean){  
 RecipeModel recipeModelImpl = **new** RecipeModelImpl();  
 recipeModelImpl.updateRecipe(recipeBean);  
 }  
   
}

**package** com.ChildHealthDiet.app2.presenter;  
  
**import** android.content.Context;  
**import** com.ChildHealthDiet.app2.context.Kidinfo;  
**import** com.ChildHealthDiet.app2.context.UserContext;  
**import** com.ChildHealthDiet.app2.model.MonthRecipeModelImpl;  
**import** com.ChildHealthDiet.app2.model.RecipeModelImpl;  
**import** com.ChildHealthDiet.app2.model.bean.MonthRecipe;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeBean;  
**import** com.ChildHealthDiet.app2.model.contract.MonthRecipeModel;  
**import** com.ChildHealthDiet.app2.model.contract.RecipeModel;  
**import** com.ChildHealthDiet.app2.presenter.contract.RecipeKeywordListContract;  
**import** com.ChildHealthDiet.app2.ui.base.RxPresenter;  
**import** com.ChildHealthDiet.app2.ui.categorys.RECIPETYPE;  
**import** java.util.List;  
**import** io.reactivex.Single;  
**import** io.reactivex.SingleEmitter;  
**import** io.reactivex.SingleOnSubscribe;  
**import** io.reactivex.android.schedulers.AndroidSchedulers;  
**import** io.reactivex.disposables.Disposable;  
**import** io.reactivex.functions.Consumer;  
**import** io.reactivex.schedulers.Schedulers;  
  
**public class** RecipeKeywordListPresenter **extends** RxPresenter<RecipeKeywordListContract.View>  
 **implements** RecipeKeywordListContract.Presenter {  
  
 RECIPETYPE **recipeType** = RECIPETYPE.***None***;  
 @Override  
 **public void** loadRecipeBeanbyMonth(Context context,String strKey){  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<List<RecipeBean>>() {  
 @Override  
 **public void** subscribe(SingleEmitter<List<RecipeBean>> emitter) **throws** Exception {  
 RecipeModel recipeModelImpl = **new** RecipeModelImpl();  
 emitter.onSuccess(recipeModelImpl.loadRecipeBeanbyMonth(context,strKey));  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<List<RecipeBean>>() {  
 @Override  
 **public void** accept(List<RecipeBean> recipeBeans) **throws** Exception {  
 RecipeKeywordListPresenter.**this**.getView().updateRecipe(recipeBeans);  
 }  
 });  
 addDisposable(disposable);  
 }  
 @Override  
 **public void** loadRecipeByKeyword(Context context,String strKey) {  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<List<RecipeBean>>() {  
 @Override  
 **public void** subscribe(SingleEmitter<List<RecipeBean>> emitter) **throws** Exception {  
 RecipeModel recipeModel = **new** RecipeModelImpl();  
 List<RecipeBean> recipeBeans = recipeModel.searchRecipeBeanbyKeyword(context,strKey);  
 emitter.onSuccess(recipeBeans);  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<List<RecipeBean>>() {  
 @Override  
 **public void** accept(List<RecipeBean> recipeBeans) **throws** Exception {  
 RecipeKeywordListPresenter.**this**.getView().updateRecipe(recipeBeans);  
 }  
 });  
 addDisposable(disposable);  
 }  
 @Override  
 **public void** filterByKeyword(Context context,List<Long> ids,String strKey){  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<List<RecipeBean>>() {  
 @Override  
 **public void** subscribe(SingleEmitter<List<RecipeBean>> emitter) **throws** Exception {  
 RecipeModel recipeModel = **new** RecipeModelImpl();  
 List<RecipeBean> recipeBeans = recipeModel.filterRecipeBeanbyKeyword(context,ids,strKey);  
 emitter.onSuccess(recipeBeans);  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<List<RecipeBean>>() {  
 @Override  
 **public void** accept(List<RecipeBean> recipeBeans) **throws** Exception {  
 RecipeKeywordListPresenter.**this**.getView().updateRecipe(recipeBeans);  
 }  
 });  
 addDisposable(disposable);  
 }  
 @Override  
 **public void** loadCollectRecipe(){  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<List<RecipeBean>>() {  
 @Override  
 **public void** subscribe(SingleEmitter<List<RecipeBean>> emitter) **throws** Exception {  
 RecipeModel recipeModel = **new** RecipeModelImpl();  
 List<RecipeBean> recipeBeans = recipeModel.getColletRecipeBean();  
 emitter.onSuccess(recipeBeans);  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<List<RecipeBean>>() {  
 @Override  
 **public void** accept(List<RecipeBean> recipeBeans) **throws** Exception {  
 RecipeKeywordListPresenter.**this**.getView().updateRecipe(recipeBeans);  
 }  
 });  
 addDisposable(disposable);  
 }  
 @Override  
 **public void** deleteCollectRecipe(List<RecipeBean> recipeBeans){  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<List<RecipeBean>>() {  
 @Override  
 **public void** subscribe(SingleEmitter<List<RecipeBean>> emitter) **throws** Exception {  
 RecipeModel recipeModel = **new** RecipeModelImpl();  
 recipeModel.deleteCollectRecipe(recipeBeans);  
 List<RecipeBean> recipeBeans2 = recipeModel.getColletRecipeBean();  
 emitter.onSuccess(recipeBeans2);  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<List<RecipeBean>>() {  
 @Override  
 **public void** accept(List<RecipeBean> recipeBeans2) **throws** Exception {  
 RecipeKeywordListPresenter.**this**.getView().updateRecipe(recipeBeans2);  
 }  
 });  
 addDisposable(disposable);  
 }  
  
  
 @Override  
 **public void** loadSymptomsRecipeBean(String strSymptoms){  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<List<RecipeBean>>() {  
 @Override  
 **public void** subscribe(SingleEmitter<List<RecipeBean>> emitter) **throws** Exception {  
 RecipeModel recipeModel = **new** RecipeModelImpl();  
 List<RecipeBean> recipeBeans = recipeModel.getSymptomsRecipeBean(strSymptoms);  
 emitter.onSuccess(recipeBeans);  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<List<RecipeBean>>() {  
 @Override  
 **public void** accept(List<RecipeBean> recipeBeans) **throws** Exception {  
 RecipeKeywordListPresenter.**this**.getView().updateRecipe(recipeBeans);  
 }  
 });  
 addDisposable(disposable);  
 }  
  
 **private** String getMonthKey(Context context,**int** intMonth){  
 String strRet = **""**;  
 MonthRecipeModel monthRecipeModel = **new** MonthRecipeModelImpl();  
 List<MonthRecipe> monthRecipes = monthRecipeModel.loadMonthRecipe(context);  
 **for** (**int** i=0;i<monthRecipes.size();i++){  
 MonthRecipe itemMonthRecipe = monthRecipes.get(i);**if**(intMonth < 4 || intMonth > 72){  
 **break**;  
 }  
 **if**(itemMonthRecipe.getKey().indexOf(**"-"**)>0){  
 String startMonth = itemMonthRecipe.getKey().split(**"-"**)[0];  
 String endMonth = itemMonthRecipe.getKey().split(**"-"**)[1];  
  
 **int** intStartMonth = **this**.convertStrtoint(startMonth);  
 **int** intEndMonth = **this**.convertStrtoint(endMonth);  
 **if**(intMonth>=intStartMonth && intMonth<=intEndMonth){  
 strRet = itemMonthRecipe.getKey();  
 **break**;  
 }  
 }  
 **else**{  
 String curMonth = itemMonthRecipe.getKey();  
 **int** intcurMonth = **this**.convertStrtoint(curMonth);  
 **if**(intMonth == intcurMonth){  
 strRet = itemMonthRecipe.getKey();  
 **break**;  
 }  
 }  
 }  
 **return** strRet;  
 }  
  
 **private int** convertStrtoint(String strValue){  
 **int** b=0;  
 **try** {  
 b = Integer.*valueOf*(strValue).intValue();  
 }  
 **catch** (NumberFormatException e)  
 {  
 e.printStackTrace();  
 }  
 **return** b;  
 }  
  
 @Override  
 **public void** loadEattimeRecipeBean(Context context,String strEatTime){  
 Kidinfo kidinfo = UserContext.*getInstance*().getmKidinfo(context);  
 **int** intMonth =0;  
 **try**{  
 intMonth = kidinfo.getMonthAge();  
 }  
 **catch** (Exception e){  
 }  
 String strMonthKey = **this**.getMonthKey(context,intMonth);  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<List<RecipeBean>>() {  
 @Override  
 **public void** subscribe(SingleEmitter<List<RecipeBean>> emitter) **throws** Exception {  
 RecipeModel recipeModel = **new** RecipeModelImpl();  
 List<RecipeBean> recipeBeans = recipeModel.getEattimeRecipeBean(strEatTime,strMonthKey);  
 emitter.onSuccess(recipeBeans);  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<List<RecipeBean>>() {  
 @Override  
 **public void** accept(List<RecipeBean> recipeBeans) **throws** Exception {  
 RecipeKeywordListPresenter.**this**.getView().updateRecipe(recipeBeans);  
 }  
 });  
 addDisposable(disposable);  
 }  
 @Override  
 **public void** loadTypeRecipeBean(String strType){  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<List<RecipeBean>>() {  
 @Override  
 **public void** subscribe(SingleEmitter<List<RecipeBean>> emitter) **throws** Exception {  
 RecipeModel recipeModel = **new** RecipeModelImpl();  
 List<RecipeBean> recipeBeans = recipeModel.getTypeRecipeBean(strType);  
 emitter.onSuccess(recipeBeans);  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<List<RecipeBean>>() {  
 @Override  
 **public void** accept(List<RecipeBean> recipeBeans) **throws** Exception {  
 RecipeKeywordListPresenter.**this**.getView().updateRecipe(recipeBeans);  
 }  
 });  
 addDisposable(disposable);  
 }  
 @Override  
 **public void** loadIngredientsRecipeBean(String strIngredients){  
 Disposable disposable =  
 Single.*create*(**new** SingleOnSubscribe<List<RecipeBean>>() {  
 @Override  
 **public void** subscribe(SingleEmitter<List<RecipeBean>> emitter) **throws** Exception {  
 RecipeModel recipeModel = **new** RecipeModelImpl();  
 List<RecipeBean> recipeBeans = recipeModel.getIngredientsRecipeBean(strIngredients);  
 emitter.onSuccess(recipeBeans);  
 }  
 }).subscribeOn(Schedulers.*io*())  
 .observeOn(AndroidSchedulers.*mainThread*())  
 .subscribe(**new** Consumer<List<RecipeBean>>() {  
 @Override  
 **public void** accept(List<RecipeBean> recipeBeans) **throws** Exception {  
 RecipeKeywordListPresenter.**this**.getView().updateRecipe(recipeBeans);  
 }  
 });  
 addDisposable(disposable);  
 }  
}

**package** com.ChildHealthDiet.app2.ui.activitys;  
  
**import** android.content.Context;  
**import** android.content.DialogInterface;  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.support.v7.app.AlertDialog;  
**import** android.support.v7.widget.LinearLayoutManager;  
**import** android.support.v7.widget.Toolbar;  
**import** android.view.View;  
**import** com.ChildHealthDiet.app2.R;  
**import** com.ChildHealthDiet.app2.adapter.RecipeBasketDataAdapter;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeBean;  
**import** com.ChildHealthDiet.app2.presenter.BasketPresenter;  
**import** com.ChildHealthDiet.app2.presenter.contract.BasketContract;  
**import** com.ChildHealthDiet.app2.ui.base.BaseMVPActivity;  
**import** com.github.jdsjlzx.interfaces.OnItemClickListener;  
**import** com.github.jdsjlzx.recyclerview.LRecyclerView;  
**import** com.github.jdsjlzx.recyclerview.LRecyclerViewAdapter;  
**import** java.util.List;  
**import** butterknife.BindView;  
**import** butterknife.OnClick;  
  
**public class** BasketActivity **extends** BaseMVPActivity<BasketContract.Presenter>  
 **implements** BasketContract.View {  
  
 @BindView(R.id.***basket\_recycler\_view***)  
 LRecyclerView **mRecyclerView**;  
 **private** LRecyclerViewAdapter **mLRecyclerViewAdapter** = **null**;  
 **private** RecipeBasketDataAdapter **mRecipeBasketDataAdapter** = **null**;  
 @BindView(R.id.***basket\_toolbar\_id***)  
 Toolbar **mToolbar**;  
  
 **public static void** startActivity(Context context) {  
 Intent intent = **new** Intent(context, BasketActivity.**class**);  
 context.startActivity(intent);  
 }  
 @Override  
 **protected int** getContentId() {  
 **return** R.layout.***activity\_basket\_recipe***;  
 }

@Override  
 **public void** showError() {  
 }

@Override  
 **public void** complete() {  
 }

@Override  
 **protected void** setUpToolbar(Toolbar toolbar) {  
 }  
 @Override  
 **protected void** initData(Bundle savedInstanceState) {  
 **super**.initData(savedInstanceState);  
 }@Override  
 **protected void** initWidget() {  
 initLRecylerView();  
 }@Override  
 **protected void** initClick() {  
 **mLRecyclerViewAdapter**.setOnItemClickListener(**new** OnItemClickListener() {  
 @Override  
 **public void** onItemClick(View view, **int** position) {  
 }  
 });  
 }  
 @Override  
 **protected** BasketContract.Presenter bindPresenter() {  
 **return new** BasketPresenter();  
 }

@Override  
 **protected void** processLogic() {  
 **super**.processLogic();  
 **mPresenter**.getBasketRecipeList();  
 }  
 **private void** initLRecylerView() {  
 **mRecipeBasketDataAdapter** = **new** RecipeBasketDataAdapter(**this**);  
 **mLRecyclerViewAdapter** = **new** LRecyclerViewAdapter(**mRecipeBasketDataAdapter**);  
 **mRecyclerView**.setAdapter(**mLRecyclerViewAdapter**);  
 **mRecyclerView**.setLayoutManager(**new** LinearLayoutManager(**this**));**mRecyclerView**.setPullRefreshEnabled(**false**);**mRecyclerView**.setLoadMoreEnabled(**false**);  
   
 }  
 @Override  
 **protected void** initToolbar() {  
 setSupportActionBar(**mToolbar**);  
 getSupportActionBar().setTitle(**""**);  
  
 **mToolbar**.setNavigationOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 finish();  
 }  
 });  
 }  
 @Override  
 **public void** updateBasketRecipeList(List<RecipeBean> recipeBeans) {  
 **mRecipeBasketDataAdapter**.setDataList(recipeBeans);  
 **mLRecyclerViewAdapter**.notifyDataSetChanged();  
 }  
 @OnClick(R.id.***basket\_clear\_toolbar\_id***)  
 **protected void** onClickClearButton() {  
  
 List<RecipeBean> recipeBeans = **mRecipeBasketDataAdapter**.getDataList();  
 **if** (recipeBeans == **null** || recipeBeans.size() == 0) {  
 **return**;  
 }  
 RecipeBean[] aryRecipeBean = **new** RecipeBean[recipeBeans.size()];  
 **for** (**int** i = 0; i < recipeBeans.size(); i++) {  
 recipeBeans.get(i).setBasket(**false**);  
 aryRecipeBean[i] = recipeBeans.get(i);  
 }  
  
 **final** AlertDialog.Builder normalDialog = **new** AlertDialog.Builder(BasketActivity.**this**);  
normalDialog.setTitle(**"提示"**);  
 normalDialog.setMessage(**"你确定要点删除这些吗?"**);  
 normalDialog.setPositiveButton(**"确定"**,  
 **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {BasketActivity.**this**.**mPresenter**.updateRecipes(aryRecipeBean);  
 }  
 });  
 normalDialog.setNegativeButton(**"取消"**,  
 **new** DialogInterface.OnClickListener() {  
 @Override  
 **public void** onClick(DialogInterface dialog, **int** which) {  
 *//...To-do* dialog.dismiss();  
 }  
 });AlertDialog dialog = normalDialog.create();  
 *// 显示* dialog.show();  
 dialog.getButton(dialog.***BUTTON\_POSITIVE***).setTextColor(**this**.getResources().getColor(R.color.***colorPrimaryDark***));  
 dialog.getButton(dialog.***BUTTON\_NEGATIVE***).setTextColor(**this**.getResources().getColor(R.color.***colorPrimaryDark***));  
 dialog.setCanceledOnTouchOutside(**false**);*//禁止点击 dialog 外部取消弹窗* }  
}

**package** com.ChildHealthDiet.app2.utils;  
**import** android.annotation.SuppressLint;  
**import** android.support.design.internal.BottomNavigationItemView;  
**import** android.support.design.internal.BottomNavigationMenuView;  
**import** android.support.design.widget.BottomNavigationView;  
**import** android.widget.TextView;  
**import** java.lang.reflect.Field;  
**public class** BottomNavigationViewHelper {  
 **public static void** disableShiftMode(BottomNavigationView navigationView) {  
 BottomNavigationMenuView menuView = (BottomNavigationMenuView) navigationView.getChildAt(0);  
 **try** {  
 Field shiftingMode = menuView.getClass().getDeclaredField(**"mShiftingMode"**);  
 shiftingMode.setAccessible(**true**);  
 shiftingMode.setBoolean(menuView, **false**);  
 shiftingMode.setAccessible(**false**);  
 **for** (**int** i = 0; i < menuView.getChildCount(); i++) {  
 BottomNavigationItemView itemView = (BottomNavigationItemView) menuView.getChildAt(i);  
 itemView.setShiftingMode(**false**);  
 itemView.setChecked(itemView.getItemData().isChecked());  
 }  
 } **catch** (NoSuchFieldException | IllegalAccessException e) {  
 e.printStackTrace();  
 }  
 }  
   
}

**package** com.ChildHealthDiet.app2.searchmlist;  
  
**import** android.content.Context;  
**import** android.util.AttributeSet;  
**import** android.view.ViewGroup;  
**import** android.widget.GridView;  
  
**public class** selfSearchGridView **extends** GridView   
 **public** selfSearchGridView(Context context) {  
 **super**(context);  
 }  
 **public** selfSearchGridView(Context context, AttributeSet attrs) {  
 **super**(context, attrs);  
 }  
 **public** selfSearchGridView(Context context, AttributeSet attrs, **int** defStyleAttr) {  
 **super**(context, attrs, defStyleAttr);  
 }  
 @Override  
 **protected void** onMeasure(**int** widthMeasureSpec, **int** heightMeasureSpec) {  
 **int** heightSpec;  
 **if**(getLayoutParams().**height**== ViewGroup.LayoutParams.***WRAP\_CONTENT***)  
 {  
 heightSpec = MeasureSpec.*makeMeasureSpec*(  
 Integer.***MAX\_VALUE*** >> 2, MeasureSpec.***AT\_MOST***);}  
 **else** {heightSpec = heightMeasureSpec;  
 }  
 **super**.onMeasure(widthMeasureSpec, heightSpec);  
 }  
}

**package** com.ChildHealthDiet.app2.searchmlist;  
  
**import** android.content.Context;  
**import** android.content.res.TypedArray;  
**import** android.database.Cursor;  
**import** android.database.sqlite.SQLiteDatabase;  
**import** android.graphics.Color;  
**import** android.graphics.drawable.ColorDrawable;  
**import** android.text.Editable;  
**import** android.text.TextWatcher;  
**import** android.util.AttributeSet;  
**import** android.view.KeyEvent;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.view.inputmethod.EditorInfo;  
**import** android.widget.AdapterView;  
**import** android.widget.Button;  
**import** android.widget.CursorAdapter;  
**import** android.widget.EditText;  
**import** android.widget.ImageView;  
**import** android.widget.LinearLayout;  
**import** android.widget.SimpleCursorAdapter;  
**import** android.widget.TextView;  
**import** java.util.ArrayList;  
**import** java.util.List;  
**import** java.util.Random;  
  
**public class** mSearchLayout **extends** LinearLayout {  
  
 **private** String **msearch\_hint**;  
 **private int msearch\_baground**;  
 Context **context**;  
 **private** ImageView **ib\_searchtext\_delete**;  
 **private** EditText **et\_searchtext\_search**;  
 **private** LinearLayout **searchview**;  
 **private** Button **bt\_text\_search\_back**;  
 **private** TextView **tvclearolddata**;  
 *//历史搜索* **private** selfSearchGridView **gridviewolddata**;  
 **private** SearchOldDataAdapter **OldDataAdapter**;  
 **private** ArrayList<String> **OldDataList** = **new** ArrayList<String>();  
 *//热门搜索* FlowLayout **hotflowLayout**;  
 **private** String **backtitle** = **"取消"**, **searchtitle** = **"搜索"**;  
 **private** OnClickListener **TextViewItemListener**;  
 **private int countOldDataSize** = 15;*//默认搜索记录的条数， 正确的是传入进来的条数* **private** RecordSQLiteOpenHelper **helper**;  
 **private** SQLiteDatabase **db**;  
  
 **public** mSearchLayout(Context context) {  
 **super**(context);  
 **this**.**context** = context;  
 InitView();  
 }  
  
 **public** mSearchLayout(Context context, AttributeSet attrs) {  
 **super**(context, attrs);  
 **this**.**context** = context;  
 TypedArray ta = context.obtainStyledAttributes(attrs, R.styleable.*searchmlist*);  
 **msearch\_hint** = ta.getString(R.styleable.*searchmlist\_search\_hint*);  
 **msearch\_baground** = ta.getResourceId(R.styleable.*searchmlist\_search\_baground*, R.drawable.*search\_baground\_shap*);  
 ta.recycle();  
 InitView();  
 }  
 **public** mSearchLayout(Context context, AttributeSet attrs, **int** defStyleAttr) {  
 **super**(context, attrs, defStyleAttr);  
 **this**.**context** = context;  
 InitView();  
 }

**private void** InitView() {  
 **backtitle** = getResources().getString(R.string.*search\_cancel*);  
 **searchtitle** = getResources().getString(R.string.*search\_verify*);  
 **searchview** = (LinearLayout) LayoutInflater.*from*(**context**).inflate(R.layout.*msearchlayout*, **null**);addView(**searchview**);**ib\_searchtext\_delete** = (ImageView) **searchview**.findViewById(R.id.*ib\_searchtext\_delete*);  
 **et\_searchtext\_search** = (EditText) **searchview**.findViewById(R.id.*et\_searchtext\_search*);  
 **et\_searchtext\_search**.setBackgroundResource(**msearch\_baground**);  
 **et\_searchtext\_search**.setHint(**msearch\_hint**);  
 *//搜索返回时一个按钮* **bt\_text\_search\_back** = (Button) **searchview**.findViewById(R.id.*buttonback*);**tvclearolddata** = (TextView) **searchview**.findViewById(R.id.*tvclearolddata*);  
 **gridviewolddata** = (selfSearchGridView) **searchview**.findViewById(R.id.*gridviewolddata*);  
 **gridviewolddata**.setSelector(**new** ColorDrawable(Color.***TRANSPARENT***));**hotflowLayout** = (FlowLayout) **searchview**.findViewById(R.id.*id\_flowlayouthot*);  
 setLinstener();  
  
 }  
 *//文本观察者* **private class** MyTextWatcher **implements** TextWatcher {  
 @Override  
 **public void** afterTextChanged(Editable s) {}  
 @Override  
 **public void** beforeTextChanged(CharSequence s, **int** start, **int** count,  
 **int** after) {}@Override  
 **public void** onTextChanged(CharSequence s, **int** start, **int** before,  
 **int** count) {*//如果编辑框中文本的长度大于0就显示删除按钮否则不显示* **if** (s.length() > 0) {  
 **ib\_searchtext\_delete**.setVisibility(View.***VISIBLE***);  
 **bt\_text\_search\_back**.setText(**searchtitle**);  
 } **else** {  
 **ib\_searchtext\_delete**.setVisibility(View.***GONE***);  
 **bt\_text\_search\_back**.setText(**backtitle**);  
 }  
 }  
 }

**private void** setLinstener() {  
 *//给删除按钮添加点击事件* **ib\_searchtext\_delete**.setOnClickListener(**new** OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 *//* ***TODO Auto-generated method stub* et\_searchtext\_search**.setText(**""**);  
 }  
 });

*//给编辑框添加文本改变事件* **et\_searchtext\_search**.addTextChangedListener(**new** MyTextWatcher());  
 **et\_searchtext\_search**.setOnEditorActionListener(**new** TextView.OnEditorActionListener() {  
 **public boolean** onEditorAction(TextView v, **int** actionId, KeyEvent event) {  
 **if** (actionId == EditorInfo.***IME\_ACTION\_SEARCH*** || (event != **null** && event.getKeyCode() == KeyEvent.***KEYCODE\_ENTER***)) {  
  
 String searchtext = **et\_searchtext\_search**.getText().toString().trim();executeSearch\_and\_NotifyDataSetChanged(searchtext);  
 **return true**;  
 }  
 **return false**;  
 }  
 });  
  
 **bt\_text\_search\_back**.setOnClickListener(**new** OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 String searchtext = **et\_searchtext\_search**.getText().toString().trim();  
 **if** (**bt\_text\_search\_back**.getText().toString().equals(**searchtitle**)) {executeSearch\_and\_NotifyDataSetChanged(searchtext);  
 } **else** {**if** (**sCBlistener** != **null**)  
 **sCBlistener**.Back();  
 }  
 }  
 });  
 **tvclearolddata**.setOnClickListener(**new** OnClickListener() {  
 @Override  
 **public void** onClick(View v) {**if** (**sCBlistener** != **null**) {  
 mSearchLayout.**this**.deleteData();  
 **OldDataList**.clear();  
 **OldDataAdapter**.notifyDataSetChanged();  
 **sCBlistener**.ClearOldData();  
 }  
 }  
 });  
 **TextViewItemListener** = **new** OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 String string = ((TextView) v).getText().toString();executeSearch\_and\_NotifyDataSetChanged(string);  
 }  
 };  
  
 **gridviewolddata**.setOnItemClickListener(**new** AdapterView.OnItemClickListener() {  
 @Override  
 **public void** onItemClick(AdapterView<?> parent, View view, **int** position, **long** id) {**if** (**sCBlistener** != **null**) {  
 **sCBlistener**.Search(**OldDataList**.get(position).trim());  
 }  
 }  
 });  
 }  
**public void** initData(List<String> olddatalist, List<String> hotdata, setSearchCallBackListener sCb) {  
 SetCallBackListener(sCb);  
 **hotflowLayout**.removeAllViews();  
 loaddb();  
 **OldDataList**.clear();olddatalist = queryData(**""**);  
 **OldDataList**.addAll(olddatalist);**OldDataAdapter** = **new** SearchOldDataAdapter(**context**, **OldDataList**);  
 **gridviewolddata**.setAdapter(**OldDataAdapter**);  
 LayoutInflater mInflater = LayoutInflater.*from*(**context**);  
 **for** (**int** i = 0; i < hotdata.size(); i++) {  
 TextView tv = (TextView) mInflater.inflate(R.layout.*suosou\_item*, **hotflowLayout**, **false**);  
 tv.setText(hotdata.get(i));  
 tv.setOnClickListener(**TextViewItemListener**);  
 tv.getBackground().setLevel(MyRandom(1, 5));  
 **hotflowLayout**.addView(tv);  
 }  
 }  
  
 **private void** executeSearch\_and\_NotifyDataSetChanged(String str) {  
 **if** (**sCBlistener** != **null** && (!str.equals(**""**))) {  
 **if** (**OldDataList**.size() > 0 && **OldDataList**.get(0).equals(str)) {  
 } **else** {  
 **if** (**OldDataList**.size() == **countOldDataSize** && **OldDataList**.size() > 0)  
 **OldDataList**.remove(**OldDataList**.size() - 1);**sCBlistener**.SaveOldData(**OldDataList**);  
 **this**.insertKeyword(str);  
 }  
 **sCBlistener**.Search(str);  
 }  
 }**private void** loaddb() {**helper** = **new** RecordSQLiteOpenHelper(**context**);queryData(**""**);  
 }  
 **private void** insertKeyword(String keyword){  
 **boolean** hasData = hasData(keyword.trim());**if** (!hasData) {  
 insertData(keyword.trim());  
 }  
 **else**{  
 deleteData(keyword.trim());  
 insertData(keyword.trim());  
 }  
 **OldDataList**.clear();  
 List<String> keywords = queryData(**""**);  
 **OldDataList**.addAll(keywords);  
 **OldDataAdapter**.notifyDataSetChanged();  
 }  
 **private void** clearKeyword(){  
 deleteData();  
 **OldDataList**.clear();  
 **OldDataAdapter**.notifyDataSetChanged();  
 }**private** List<String> queryData(String tempName) {Cursor cursor = **helper**.getReadableDatabase().rawQuery(  
 **"select id as \_id,name from records where name like '%"** + tempName + **"%' order by id desc "**, **null**);List<String> namelist = **new** ArrayList<String>();  
 **if** (cursor != **null** && cursor.getCount() > 0) {  
 **while** (cursor.moveToNext()) {*//游标是否向下移动* String strItem = cursor.getString(cursor.getColumnIndex(**"name"**));  
 namelist.add(strItem);  
 }  
 *//关闭cursor* cursor.close();  
 }  
 **return** namelist;  
 }**private void** deleteData() {  
  
 **db** = **helper**.getWritableDatabase();  
 **db**.execSQL(**"delete from records"**);  
 **db**.close();  
 }**private boolean** hasData(String tempName) {Cursor cursor = **helper**.getReadableDatabase().rawQuery(  
 **"select id as \_id,name from records where name =?"**, **new** String[]{tempName});**return** cursor.moveToNext();  
 }  
 **private void** deleteData(String tempName) {  
 *// 从数据库中Record表里找到name=tempName的id* **db** = **helper**.getWritableDatabase();  
 **db**.execSQL(**"delete from records where name =?"**, **new** String[]{tempName});  
 **db**.close();  
 }  
}

**package** com.ChildHealthDiet.app2.ui.activitys;  
  
**import** android.app.AlertDialog;  
**import** android.app.DatePickerDialog;  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.support.v7.widget.Toolbar;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.DatePicker;  
**import** android.widget.EditText;  
**import** com.ChildHealthDiet.app2.R;  
**import** com.ChildHealthDiet.app2.context.Kidinfo;  
**import** com.ChildHealthDiet.app2.context.UserContext;  
**import** com.ChildHealthDiet.app2.ui.base.BaseActivity;  
**import** java.util.Date;  
**import** butterknife.BindView;  
**import** butterknife.OnClick;  
  
**public class** KidEidtActivity **extends** BaseActivity {  
 @BindView(R.id.***edittext\_nickname***)  
 **protected** EditText **editTextNickname**;  
 @BindView(R.id.***btn\_pick\_age***)  
 **protected** Button **btnKidAge**;  
 @BindView(R.id.***mine\_edit\_toolbar\_id***)  
 Toolbar **mToolbar**;  
 **int mYear**=2017,**mMonth**=01,**mDay**=01;  
 @Override  
 **protected int** getContentId() {  
 **return** R.layout.***activity\_kid\_eidt***;  
 }  
 @Override  
 **protected void** initWidget() {  
 loadData();  
 }  
 **public static void** startActivity(android.support.v4.app.Fragment fragment){fragment.startActivityForResult(**new** Intent(fragment.getContext(), KidEidtActivity.**class**), 10030);  
 }  
 **private void** initEvent(){  
  
 }  
 **protected void** loadData(){  
 Kidinfo kidinfo = UserContext.*getInstance*().getmKidinfo(**this**);  
 **if**(!kidinfo.getNickName().equals(**""**)){  
 **this**.**editTextNickname**.setText(kidinfo.getNickName());  
 }  
 **if**(!kidinfo.getBirthdate().equals(**""**)){  
 **this**.**btnKidAge**.setText(kidinfo.getBirthdate());  
 }  
 }  
 @OnClick({R.id.***btn\_pick\_age***})  
 **protected void** onClickAgeButton(){  
 DatePickerDialog datePickerDialog = **new** DatePickerDialog(KidEidtActivity.**this**,AlertDialog.***THEME\_HOLO\_LIGHT***,  
 **new** DatePickerDialog.OnDateSetListener() {  
 @Override  
 **public void** onDateSet(DatePicker view, **int** year, **int** month, **int** dayOfMonth) {  
 **mYear** = year;  
 **mMonth** = month;  
 **mDay** = dayOfMonth;  
  
 **final** String data = year +**"-"** + (month+1<10?**"0"**+(month+1):month+1) + **"-"** + (dayOfMonth<10?**"0"**+dayOfMonth:dayOfMonth) + **""**;  
KidEidtActivity.**this**.**btnKidAge**.setText(data);  
 }  
 },  
 **mYear**, **mMonth**, **mDay**);  
 **if** (android.os.Build.VERSION.***SDK\_INT***>=11) {  
  
 DatePicker dp = datePickerDialog.getDatePicker();  
  
 dp.setMaxDate(**new** Date().getTime());  
  
 }  
 datePickerDialog.show();  
 }  
 @Override  
 **protected void** onCreate(Bundle savedInstanceState) {  
 **super**.onCreate(savedInstanceState);  
 }  
 @Override  
 **protected void** initToolbar() {  
 setSupportActionBar(**mToolbar**);  
 getSupportActionBar().setTitle(**""**);  
 **mToolbar**.setNavigationOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 finish();  
 }  
 });  
 }  
 @Override  
 **protected void** onPause(){  
 **super**.onPause();  
 **this**.saveData();  
  
 }  
 **private void** saveData() {  
 UserContext.*getInstance*().setmKidinfo(**this**,**this**.**editTextNickname**.getText().toString(),  
 **this**.**btnKidAge**.getText().toString().equals(**"请输入出生日期"**)?**""**:**this**.**btnKidAge**.getText().toString());  
 }  
 @Override  
 **public void** onBackPressed() {  
 **boolean** success = getSupportFragmentManager().popBackStackImmediate();  
 **if** (!success)  
 **super**.onBackPressed();  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
 **int** id = item.getItemId();  
 **if** (id == android.R.id.***home***) {  
 finish();  
 }  
  
 **return super**.onOptionsItemSelected(item);  
 }  
  
}

**package** com.ChildHealthDiet.app2.ui.activitys;  
  
**import** android.app.Activity;  
**import** android.content.Intent;  
**import** android.os.Bundle;  
**import** android.support.design.widget.CollapsingToolbarLayout;  
**import** android.support.v4.app.ActivityOptionsCompat;  
**import** android.support.v4.util.Pair;  
**import** android.support.v7.app.ActionBar;  
**import** android.support.v7.widget.LinearLayoutManager;  
**import** android.support.v7.widget.RecyclerView;  
**import** android.support.v7.widget.Toolbar;

**import** android.util.Log;  
**import** android.view.MenuItem;  
**import** android.view.View;  
**import** android.view.ViewGroup;  
**import** android.widget.ImageButton;  
**import** android.widget.ImageView;**import** com.ChildHealthDiet.app2.Constants;  
**import** com.ChildHealthDiet.app2.ConstantsAdmob;  
**import** com.bumptech.glide.Glide;

**import** com.ChildHealthDiet.app2.R;  
**import** com.ChildHealthDiet.app2.adapter.RecipeDetailAdapter;  
**import** com.ChildHealthDiet.app2.model.bean.RecipeBean;  
**import** com.ChildHealthDiet.app2.presenter.RecipeDetailPresenter;  
**import** com.ChildHealthDiet.app2.presenter.contract.RecipeDetailContract;  
**import** com.ChildHealthDiet.app2.ui.base.BaseMVPActivity;  
**import** com.ChildHealthDiet.app2.ui.components.StatusBarUtil;

**import** com.google.android.gms.ads.AdListener;  
**import** com.google.android.gms.ads.AdRequest;  
**import** com.google.android.gms.ads.AdSize;  
**import** com.google.android.gms.ads.AdView;  
**import** com.qq.e.ads.banner.ADSize;  
**import** com.qq.e.ads.banner.AbstractBannerADListener;  
**import** com.qq.e.ads.banner.BannerView;  
**import** com.qq.e.comm.util.AdError;  
**import** java.util.Random;  
**import** butterknife.BindView;  
**public class** RecipeDetailActivity **extends** BaseMVPActivity<RecipeDetailContract.Presenter>  
 **implements** RecipeDetailContract.View {  
 @BindView(R.id.***toolbar***)  
 **public** Toolbar **toolbar**;  
 @BindView(R.id.***toolbar\_layout***)  
 **public** CollapsingToolbarLayout **toolbarLayout**;  
 @BindView(R.id.***recyclerview\_list***)  
 **public** RecyclerView **recyclerList**;  
 @BindView(R.id.***imgv\_bg***)  
 **public** ImageView **imgvBg**;  
 @BindView(R.id.***btnShare***)  
 **public** ImageButton **imbtnShare**;  
 **private long recipeId**;  
 **private boolean isShowCollection**;  
 **private** RecipeDetailAdapter **cookDetailAdapter**;  
 @BindView(R.id.***bannerContainer***)  
 ViewGroup **bannerContainer**;  
 BannerView **bv** = **null**;  
 AdView **adView** = **null**;  
  
 **final int tryloadadMaxTimes** = 5;  
 **int tryloadadTime** = 0;  
 **private** RecipeBean **mRecipeBean**;  
 @Override  
 **protected int** getContentId() {  
 **return** R.layout.***activity\_recipe\_detail***;  
 }  
   
 @Override  
 **protected void** initData(Bundle savedInstanceState){  
   
 }  
 @Override  
 **protected void** initToolbar(){  
 StatusBarUtil.*setImmersiveStatusBar*(**this**);  
 StatusBarUtil.*setImmersiveStatusBarToolbar*(**toolbar**, **this**);  
 setSupportActionBar(**toolbar**);  
 ActionBar actionBar = getSupportActionBar();  
 **if** (actionBar != **null**)  
 actionBar.setDisplayHomeAsUpEnabled(**true**);  
 **toolbarLayout**.setCollapsedTitleTextColor(getResources().getColor(R.color.***black\_text***));  
 }@Override  
 **protected void** initWidget() {  
 }@Override  
 **protected void** initClick(){  
 }@Override  
 **protected void** processLogic(){  
 **super**.processLogic();  
 Intent intent = getIntent();  
 **long** recipeId = intent.getLongExtra(***Intnet\_Recipe\_id***,0);  
 **mPresenter**.getRecipeById(**this**,recipeId);  
 loadads();  
 }   
 @Override  
 **public void** onBackPressed() {  
 **boolean** success = getSupportFragmentManager().popBackStackImmediate();  
 **if** (!success)  
 **super**.onBackPressed();  
 }  
  
 @Override  
 **public boolean** onOptionsItemSelected(MenuItem item) {  
 **int** id = item.getItemId();  
 **if** (id == android.R.id.***home***) {  
 finish();  
 }  
 **return super**.onOptionsItemSelected(item);  
 }  
 **private final static** String ***Intnet\_Recipe\_id*** = **"recipeId"**;  
 **private final static** String ***Intnet\_Data\_Cook*** = **"cook"**;  
 **private final static** String ***Intnet\_Data\_Collection*** = **"collection"**;  
  
 **public static void** startActivity(Activity activity, View view, **long** recipeId){  
 Intent intent = **new** Intent(activity, RecipeDetailActivity.**class**);intent.putExtra(***Intnet\_Recipe\_id***, recipeId);  
  
 ActivityOptionsCompat options =  
 ActivityOptionsCompat.*makeSceneTransitionAnimation*(activity  
 , Pair.*create*(view, activity.getString(R.string.***transition\_cook\_detail\_imgv\_bg***))  
 , Pair.*create*(view, activity.getString(R.string.***transition\_cook\_detail\_content***))  
 );  
 activity.startActivityForResult(intent, 10029, options.toBundle());  
 }  
 @Override  
 **public void** updateRecipe(RecipeBean recipeBean) {  
 String recipeUrl = **"file:///android\_asset/recipeimage/"** + recipeBean.getPicture();  
 Glide.*with*(**this**).load(recipeUrl).into(**this**.**imgvBg**);  
 getSupportActionBar().setTitle(recipeBean.getName());  
 **toolbarLayout**.setTitle(recipeBean.getName());  
 **cookDetailAdapter** = **new** RecipeDetailAdapter(**this**, recipeBean,**mPresenter**,**isShowCollection**);  
 **recyclerList**.setLayoutManager(**new** LinearLayoutManager(**this**));  
 **recyclerList**.setAdapter(**cookDetailAdapter**);  
 }  
  
 @Override  
 **public void** showError() {  
 }  
 @Override  
 **public void** complete() {  
 }  
 @Override  
 **protected** RecipeDetailContract.Presenter bindPresenter() {  
 **return new** RecipeDetailPresenter();  
 }  
 **private void** loadads(){  
 **if**(**this**.**adView** != **null** || **this**.**bv** != **null**)  
 {  
 **bannerContainer**.removeAllViews();  
 }  
 **if**(**tryloadadTime**>=**tryloadadMaxTimes**){  
 **return**;  
 }  
 **tryloadadTime**++;  
 **int** min=0;  
 **int** max=99;  
 Random random = **new** Random();  
 **int** num = random.nextInt(max)%(max-min+1) + min;  
 **if**(num<=90){  
 **this**.getBanner().loadAD();  
 }  
 **else**{  
 getAdView();  
 }   
 }  
 **private** String getPosID() {  
 **return** Constants.***BannerPosID***;  
 }  
 AdView getAdView(){  
 **adView** = **new** AdView(RecipeDetailActivity.**this**);  
 **adView**.setAdUnitId(ConstantsAdmob.***BannerPosID***);  
 **adView**.setAdListener(**new** AdListener() {  
 @Override  
 **public void** onAdLoaded() {  
 **tryloadadTime** = 0;  
 **super**.onAdLoaded();  
 }  
 @Override  
 **public void** onAdFailedToLoad(**int** errorCode) {Log.*e*(**"MainActivity"**, **"The previous banner ad failed to load. Attempting to"** + **" load the next banner ad in the items list."**);  
 loadads();  
 }  
 });  
 *// Load the banner ad.* **adView**.setAdSize(AdSize.***SMART\_BANNER***);  
 **adView**.loadAd(**new** AdRequest.Builder().build());  
 **bannerContainer**.addView(**adView**);  
 **return adView**;  
 }  
  
 **private** BannerView getBanner() {  
 String posId = getPosID();  
 **this**.**bv** = **new** BannerView(**this**, ADSize.***BANNER***, Constants.***APPID***,posId);  
 **bv**.setRefresh(30);  
 **bv**.setADListener(**new** AbstractBannerADListener() {  
 @Override  
 **public void** onNoAD(AdError error) {  
 **tryloadadTime** = 0;  
 Log.*i*(  
 **"AD\_DEMO"**,  
 String.*format*(**"Banner onNoAD，eCode = %d, eMsg = %s"**, error.getErrorCode(),  
 error.getErrorMsg()));  
 loadads();  
 }  
 @Override  
 **public void** onADReceiv() {  
 Log.*i*(**"AD\_DEMO"**, **"ONBannerReceive"**);  
 }  
 });  
 **bannerContainer**.addView(**bv**);  
 **return this**.**bv**;  
 }  
 @Override  
 **protected void** onResume() {  
 **if**(**adView**!=**null**){  
 **adView**.resume();  
 }  
 **super**.onResume();  
 }  
 @Override  
 **protected void** onPause() {  
  
 **if**(**adView**!=**null**){  
 **adView**.pause();  
 }  
 **super**.onPause();  
 }  
 @Override  
 **protected void** onDestroy() {  
 **if**(**adView**!=**null**){  
 **adView**.destroy();  
 }  
 **if**(**bv**!=**null**) {  
 **bv**.destroy();  
 }  
 **super**.onDestroy();  
 }  
}

**package** com.ChildHealthDiet.app2.ui.fragment;  
  
**import** android.content.Intent;  
**import** android.net.Uri;  
**import** android.os.Bundle;  
**import** android.support.v7.widget.LinearLayoutManager;  
**import** android.view.LayoutInflater;  
**import** android.view.View;  
**import** android.widget.Button;  
**import** android.widget.TextView;  
**import** com.ChildHealthDiet.app2.R;  
**import** com.ChildHealthDiet.app2.adapter.MineDataAdapter;  
**import** com.ChildHealthDiet.app2.context.Kidinfo;  
**import** com.ChildHealthDiet.app2.context.UserContext;  
**import** com.ChildHealthDiet.app2.model.bean.MineItem;  
**import** com.ChildHealthDiet.app2.presenter.MinePresenter;  
**import** com.ChildHealthDiet.app2.presenter.contract.MineContract;  
**import** com.ChildHealthDiet.app2.ui.activitys.BasketActivity;  
**import** com.ChildHealthDiet.app2.ui.activitys.KidEidtActivity;  
**import** com.ChildHealthDiet.app2.ui.activitys.RecipeKeywordListActivity;  
**import** com.ChildHealthDiet.app2.ui.base.BaseMVPFragment;  
**import** com.ChildHealthDiet.app2.ui.categorys.RECIPETYPE;  
**import** com.github.jdsjlzx.interfaces.OnItemClickListener;  
**import** com.github.jdsjlzx.recyclerview.LRecyclerView;  
**import** com.github.jdsjlzx.recyclerview.LRecyclerViewAdapter;  
**import** java.text.ParseException;  
**import** java.util.ArrayList;  
**import** java.util.List;  
**import** butterknife.BindView;  
  
**public class** MineFragment **extends** BaseMVPFragment<MineContract.Presenter> **implements** MineContract.View {**private static final** String ***ARG\_PARAM1*** = **"param1"**;  
 **private static final** String ***ARG\_PARAM2*** = **"param2"private** String **mParam1**;  
 **private** String **mParam2**;  
 View **headView**;  
 TextView **tvKidNickname**;  
 TextView **tvKidAge**;  
 Button **btnEdit**;  
 @BindView(R.id.***mine\_recycler\_view***)  
 LRecyclerView **mRecyclerView**;  
 **private** LRecyclerViewAdapter **mLRecyclerViewAdapter** = **null**;  
 **private** MineDataAdapter **mMineDataAdapter** = **null**;  
 **public** MineFragment() {  
 }   
 **public static** MineFragment newInstance(String param1, String param2) {  
 MineFragment fragment = **new** MineFragment();  
 Bundle args = **new** Bundle();  
 args.putString(***ARG\_PARAM1***, param1);  
 args.putString(***ARG\_PARAM2***, param2);  
 fragment.setArguments(args);  
 **return** fragment;  
 }  
 @Override  
 **protected int** getContentId() {  
 **return** R.layout.***fragment\_mine***;  
 }  
 @Override  
 **public void** updateMineItem(List<MineItem> mineItems) {  
 **mMineDataAdapter**.setDataList(mineItems);  
 **mLRecyclerViewAdapter**.notifyDataSetChanged();  
 }  
 @Override  
 **protected** MineContract.Presenter bindPresenter() {  
 **return new** MinePresenter();  
 }  
 @Override  
 **public void** showError() {  
 }  
 @Override  
 **public void** complete() {  
 }  
  
 @Override  
 **protected void** initData(Bundle savedInstanceState) {  
 **super**.initData(savedInstanceState);  
 *//loadTopBanner* }  
 @Override  
 **protected void** initWidget(Bundle savedInstanceState) {  
 **super**.initWidget(savedInstanceState);  
 **headView** = LayoutInflater.*from*(getActivity()).inflate(R.layout.***mine\_header\_view***,**null**);  
 **tvKidNickname** = **headView**.findViewById(R.id.***tv\_kid\_nickname***);  
 **tvKidAge** = **headView**.findViewById(R.id.***tv\_kid\_age***);  
 **btnEdit** = **headView**.findViewById(R.id.***btn\_eidt\_mine***);  
 initLRecylerView();  
 **mLRecyclerViewAdapter**.addHeaderView(**headView**);  
 }  
 @Override  
 **protected void** initClick() {  
 **super**.initClick();  
 **btnEdit**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View view) {  
 KidEidtActivity.*startActivity*(MineFragment.**this**);  
 }  
 });  
 **mLRecyclerViewAdapter**.setOnItemClickListener(**new** OnItemClickListener() {  
 @Override  
 **public void** onItemClick(View view, **int** position) {  
 List<MineItem> mineItems = MineFragment.**this**.**mMineDataAdapter**.getDataList();  
 String strKey = mineItems.get(position).getKey();  
 **switch** (strKey)  
 {  
 **case "collect"**: RecipeKeywordListActivity.*startActivity*(MineFragment.**this**.getContext(),RECIPETYPE.***Collect***);

**break**;  
 **case "basket"**: BasketActivity.*startActivity*(MineFragment.**this**.getContext());  
 **break**;  
 **case "feedback"**:  
 MineFragment.**this**.sendEmail();  
 **break**;  
 }  
 }  
 });  
 }  
 @Override  
 **protected void** processLogic() {  
 **super**.processLogic();  
 **mPresenter**.getMineItems(**this**.getContext());  
  
 **try** {  
 loadKidinfo();  
 } **catch** (ParseException e) {  
 e.printStackTrace();  
 }  
 }  
   
 **private void** initLRecylerView(){  
 **mMineDataAdapter** = **new** MineDataAdapter(getActivity());  
 **mLRecyclerViewAdapter** = **new** LRecyclerViewAdapter(**mMineDataAdapter**);  
 **mRecyclerView**.setAdapter(**mLRecyclerViewAdapter**);  
 **mRecyclerView**.setLayoutManager(**new** LinearLayoutManager(getActivity()));**mRecyclerView**.setPullRefreshEnabled(**false**);  
 *//禁用自动加载更多功能* **mRecyclerView**.setLoadMoreEnabled(**false**);  
 }  
  
**public void** sendEmail() {  
 Intent i = **new** Intent(Intent.***ACTION\_SEND***);i.setType(**"message/rfc822"**); *// 真机上使用这行* i.putExtra(Intent.***EXTRA\_EMAIL***,  
 **new** String[]{getString(R.string.***feedbackEmail***)});  
 String txtTitle = getResources().getString(R.string.***feedbackTitle***);  
 txtTitle = String.*format*(txtTitle, getString( R.string.***app\_name***));  
 i.putExtra(Intent.***EXTRA\_SUBJECT***, txtTitle);  
 i.putExtra(Intent.***EXTRA\_TEXT***, getString(R.string.***feedbackBody***));  
 ArrayList<Uri> uris = **new** ArrayList<>();  
 i.putParcelableArrayListExtra(Intent.***EXTRA\_STREAM***, uris);  
 startActivity(Intent.*createChooser*(i,  
 **"选择email程序."**));  
}