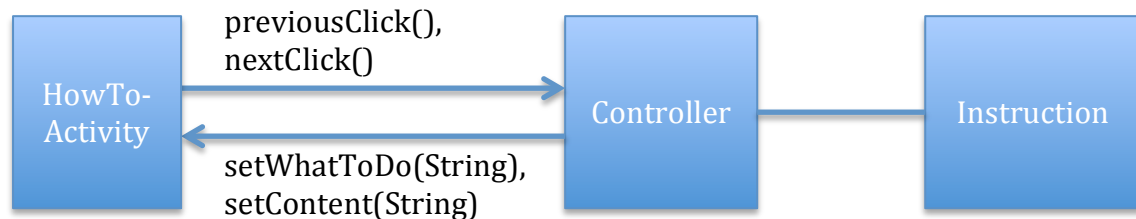


Laboration 2a

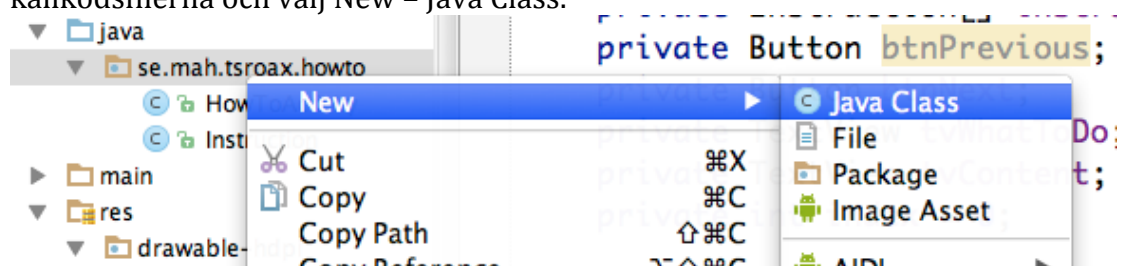
Appen HowTo, från föreläsning 1 och laboration1 ska omarbetas lite i denna uppgift. En controller-klass ska skapas vilken ska sköta logiken i programmet. UI-klassen ska endast rapportera till Controller-klassen att användaren klickat på en knapp (anrop till metoderna `previousClick()` respektive `nextClick()` i Controller-klassen) och kunna visa upp resultat som användaren ska se (Controller-klassen anropas `setWhatToDo(String)` och `setContent(String)`).



Nedan är en detaljerad beskrivning hur man kan ordna detta. Om du följer beskrivningen så måste du reflektera över förändringarna som göres.

Att göra

- Klassen **Controller** måste skapas i projektet. Högerklicka paketet med källkodsfilerna och välj New – Java Class.



- Controller-klassen ska hålla reda på Instruction-objekt och därför innehålla instansvariablerna `instructions` och `index`, vilka finns i `HowToActivity` just nu. Kopiera dem till Controller-klassen (tar bort dem lite senare ur Activity-klassen). Passa på att lägga till en instansvariabel som kan referera till `HowToActivity`-objekt och skriva en konstruktor som tar emot en referens till ett `HowToActivity`-objekt.

```
package se.mah.tsroax.howto;
```

```
public class Controller {
    private Instruction[] instructions = new Instruction[3];
    private int index = 0;
    private HowToActivity ui;

    public Controller(HowToActivity ui) {
        this.ui = ui;
    }
}
```

När `Controller`-objektet skapas så kommer en referens till `HowToActivity`-objektet som argument. Genom att lagra referensen i instansvariabeln `ui`

så kan Controller-objektet anropa metoder i HowToActivity-objektet (pilen från Controller till HowToActivity i figuren ovan).

- Skriv metoderna *previousClick* respektive *nextClick* i Controller-klassen. Metoderna blir väldigt lika händelsehanterarna i klassen HowToActivity:

```
public void previousClick() {
    index--;
    if(index<0)
        index = instructions.length-1;
    ui.setWhatToDo(instructions[index].getWhatToDo());
    ui.setContent(instructions[index].getContent());
}
```

Skillnaden mot händelsemetoden i HowToActivity är att metoden avslutas med anrop till metoderna *ui.setWhatToDo* respektive *ui.setContent*. Genom dessa anrop meddelar Controller-objektet att strängarna som skickas med som argument ska visas i Uiet. Metoderna är rödmarkerade eftersom de inte finns i klassen HowToActivity. Om du klickar på den röda texten så blir det en röd glödlampa till vänster. Klicka på denna och välj "Create method...". Metoderna skapas i HotToActivity och snart ska vi lägga till instruktioner i dem. Skriv nu metoden *nextClick* på motsvarande sätt.

- Controller-objektet måste skapa *Instruction*-objekten och placera dem i arrayen *instructions*. Detta ska göras i konstruktorn. Lägg till metodanropet *initializeResources* i konstruktorn. Gör det efter tilldelningen till *this.ui*. Värdet i *this.ui* kommer nämligen att användas i metoden. Eftersom metoden inte finns blir metodanropet rött. Men metoden finns i nästan fullgott skick i HotToActivity-klassen. Kopiera den till Controller-klassen.

```
public Controller(HowToActivity ui) {
    this.ui = ui;
    initializeResources();
}

private void initializeResources() {
    Resources res = getResources();
    String whatToDo = res.getString(R.string.what_to_do);
    String content = res.getString(R.string.content);
    instructions[0] = new Instruction(whatToDo, content);
    instructions[1] = new Instruction(res.getString(R.string.what_
    instructions[2] = new Instruction(res.getString(R.string.what_
```

Anropet till *getResources* blir rött. Det beror på att metoden *getResources* är en metod som är ärvd till klassen HowToActivity från klassen Activity. Men metoden kan anropas med hjälp av instansvariabeln *ui* (dvs *this.ui*):
`Resources res = ui.getResources();`

Nu är Controller-klassen klar och nu är det dags att anpassa **HowToActivity**-klassen. Det handlar mest om att ta bort.

- Ta bort instansvariablerna *instructions* och *index*. Det blir en del rödmarkeringar men det kommer vi åtgärda successivt. Lägg till en instansvariabel som kan referera till ett Controller-objekt.

```
public class MainActivity extends Activity {
    private Controller controller;
    private Button btnNext;
    private TextView tvWhatToDo;
    private TextView tvContent;
```

- Skapa ett *Controller*-objekt i *onCreate* och tilldela *this.controller* referens till objektet. Med hjälp av denna referens kan MainActivity-objektet meddela Controller-objektet då användaren klickar på en knapp. Du gör detta sist i metoden då initieringar av activityn är färdiga.

```
        registerListeners();
        controller = new Controller(this);
    }
```

Argumentet *this* vid konstruktionen är referens till MainActivity-objektet. Det är denna referens som Controller-objektet lagrar i instansvariabeln *ui*.

- Ta bort anropet till *initializeResources* i *onCreate*-metoden. Nu sköts detta av Controller-objektet. Ta också bort metoden *initializeResources* ur MainActivity-klassen. (3 rödmarkeringar försvann)

- Metoden *previousInstruction* ska meddela Controller-objektet att användaren klickat på Previous-knappen.

```
public void previousInstruction(View view) {
    controller.previousClick();
}
```

(5 rödmarkeringar försvann)

- Ordna händelsehanteringen vid klick på Next-knappen på motsvarande sätt.

```
private class NextListener implements View.OnClickListener {
    public void onClick(View v) {
        controller.nextClick();
    }
}
```

(inga rödmarkeringar kvar)

- Det sista som måste göras är att se till att metoderna *setWhatToDo* respektive *setContent* fser till att argumenten görs synliga.

```
public void setWhatToDo(String whatToDo) {
    tvWhatToDo.setText(whatToDo);
}

public void setContent(String content) {
    tvContent.setText(content);
}
```

Laboration 2b

Du ska tillverka en enkel applikation som består av en Activity-klass, två Fragment-klasser och en Controller-klass. Ett av Fragmenten ska innehålla en TextView-komponent och det andra Fragmentet ska innehålla en Button-komponent. TextView-komponenten ska visa antalet klick som skett på ButtonKomponenten.

Följande gäller för lösningen:

Använd en inre klass för händelshantering. Den inre klassen ska vara i Fragmentet med Button-komponenten.

Controller-klassen ska meddelas vid klick. Det innebär att Fragment-klassen med en knapp måste ha en referens till Controller-klassen.

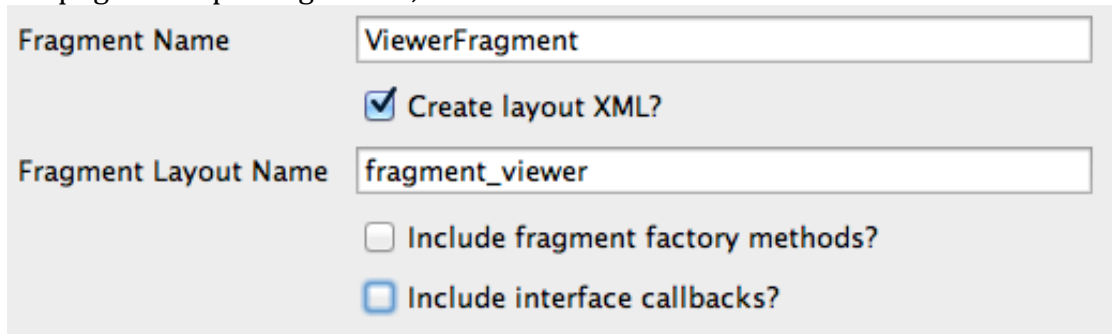
Controller-klassen ska meddela Fragmentet med en TextView vilket värde som ska visas. Ett heltal (int) görs om till ett String-objekt genom anrop till metoden `String.valueOf(int)`, t.ex.

```
int a = 10;  
String b = String.valueOf( a ); // b = "10"
```

Controller-klassen måste alltså ha referens till fragmentet med en TextView.

En tänkbar ordning (Testkör din lösning med jämna mellanum):

1. Skapa ett projekt – Blanc Activity
2. Skapa **Fragmentet** med TextView.
Höger-klicka paketet med källkod och välj: New – Fragment – Fragment (Blank). Ta bort de två nedersta markeringarna i dialogen och ange lämpligt namn på Fragmentet, t.ex.



Fragment Name	ViewerFragment
<input checked="" type="checkbox"/> Create layout XML?	
Fragment Layout Name	fragment_viewer
<input type="checkbox"/> Include fragment factory methods?	
<input type="checkbox"/> Include interface callbacks?	

TextView-komponenten placerar du i fragmentets xml-fil. Du kommer behöva referens till TextView-komponenten i din fragment-klass (se F2) och du ska lägga till en metod som Controller-objektet kan anropa för att ändra innehållet i TextView-komponenten.

3. Skapa **Controller**-klass. Lägg till lämpliga instansvariabler (2 st) och en lämplig konstruktör i klassen. Skapa en metod som kan anropas vid klick på knappen och som uppdaterar TextView-komponenten med antalet klick.
4. Skapa **Fragmentet** med Button. Button-komponenten lägger du till i fragmentets xml-fil, och tar bort TextView-komponenten. Lägg till

instansvariabler (Controller controller + komponentreferens) i klassen och lägg till en setController-metod för att instansvariabeln controller ska kunna ges ett värde

Ordna händelsehanteringen så att Controller-objektet meddelas vid klick.

5. Lägg till fragment-taggar i layouten för activityn. Välj Design-läge. I listan till vänster klickar du på <fragment>, markerar aktuell Fragment-klass, klickar på OK och klickar slutligen i mobilfönstret. Du kommer att meddelas att det är problem att rendera korrekt.

Klicka på första delen i den översta länken så löses problemet (välj Text - det understa attributet i fragment-taggen har lagts till).

Ge vettiga *id* till fragmenten.

6. Referenserna mellan objekten måste ordnas i *onCreate*-metoden i **Activity**-klassen. Något liknande koden nedan:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_lab2b);  
    initializeSystem();  
}  
  
private void initializeSystem() {  
    FragmentManager fm = getFragmentManager();  
    ViewerFragment viewer = (ViewerFragment)fm.findFragmentById(R.id.frViewer);  
    InputFragment input = (InputFragment)fm.findFragmentById(R.id.frInput);  
    Controller controller = new Controller(viewer);  
    input.setController(controller);  
}
```

Laboration 2c

Gör om HowTo-applikationen så att den innehåller Activity-klass, Fragment-klass och Controller-klass. Gör det i ett nytt projekt men återanvänd koden från Laboration 2a.

Laboration 2d

Alla bilder, xml-filer och källkodsfiler till dagens föreläsning finns på kurssidan.

Skapa ett projekt, RSPStaticFragment, bygg successivt upp appen med mina filer. Jobba helst tillsammans med någon eller några andra. Det är eventuellt ganska krävande att få ihop det, mycket kan gå fel.