



Hugbúnaðarverkefni 2 / Software Project 2

2. Agile Requirements

HBV601G – Spring 2020

Matthias Book



HÁSKÓLI ÍSLANDS
VERKFRAÐI- OG NÁTTÚRVÍSINDASVIÐ
ÍÐNAÐARVERKFRAÐI-, VÉLAVERKFRAÐI-
OG TÖLVUNARFRÆÐIDEILD

Requirements in Agile Projects

see also:

Leffingwell: Agile Software Requirements, Ch. 6, 13, 19



HÁSKÓLI ÍSLANDS

Matthias Book: Software Project 2



Recap: The Nature of Software Development

**“Because software is embodied knowledge,
and that knowledge is initially dispersed, tacit, latent, and incomplete,
software development is a social learning process.”**

Howard Baetjer, Jr.: Software as Capital. IEEE Computer Society Press, 1998



Recap: Valid Requirements Are the Key to Project Success

- The majority of project failures is due to incorrectly addressed requirements.
- Managing requirements is more challenging than technical execution of project.
- We can't rely on the customers to “just tell us” their requirements:
 - Customers often don't know exactly what they want
 - If they know what they want, they can't describe it precisely
 - If they can describe it, they describe a proposed solution rather than an actual need
 - If they describe an actual need, we still need to
 - understand it (what? how? why?)
 - explore it (variants? constraints? boundaries? exceptions? extensions? evolution?)
 - solve it (architecture? design? technology? dependencies? implementation? operation?)
- It is virtually impossible for customers or developers to have a correct and complete picture of all this right from the start of any project.

Recap: Valid Requirements Are the Key to Project Success

- It is virtually impossible to develop valid requirements up front.
- Agile purist approach: “Let’s not attempt to define precise requirements then, but figure things out along the way.”
 - Taken literally, this just surrenders to the aforementioned problems, it doesn’t solve them.
- **Caution: Agile approaches do not make requirements work easier!**
 - In particular, requirements work does not disappear just because there is no req’s document.
 - Agile approaches just interleave the requirements work more tightly with other activities and allow the team to make course corrections along the way.
 - i.e. they are in essence a risk management technique (since late corrections would not be cheap)
- **Agile approaches require investing just as much thought and effort (and discipline) into the aforementioned issues as plan-driven approaches do.**
 - “Let’s just start coding and see what the customer says” can end up in an expensive disaster just as easily as “Let’s get the complete and precise requirements from the customer first.”



Defining and Communicating the Project Vision

- In agile approaches, we strive to prevent
 - overinvesting effort in things we are unlikely to understand very well initially anyway
 - binding resources too early on a fixed set of commitments that we may wish to change later
- No explicit project definition or Software Requirements Specification is prepared
- Need another way to communicate project vision to agile development team, i.e. to make developers aware of strategic intent as overall frame of reference:
 - Why are we building this product / system / application?
 - What problem will it solve?
 - What features and benefits will it provide? For whom?
 - What platforms, standards, applications etc. will it support?
 - What system landscape will it be integrated with?
- An explicit document (can be quite concise) is still the best way to express this.

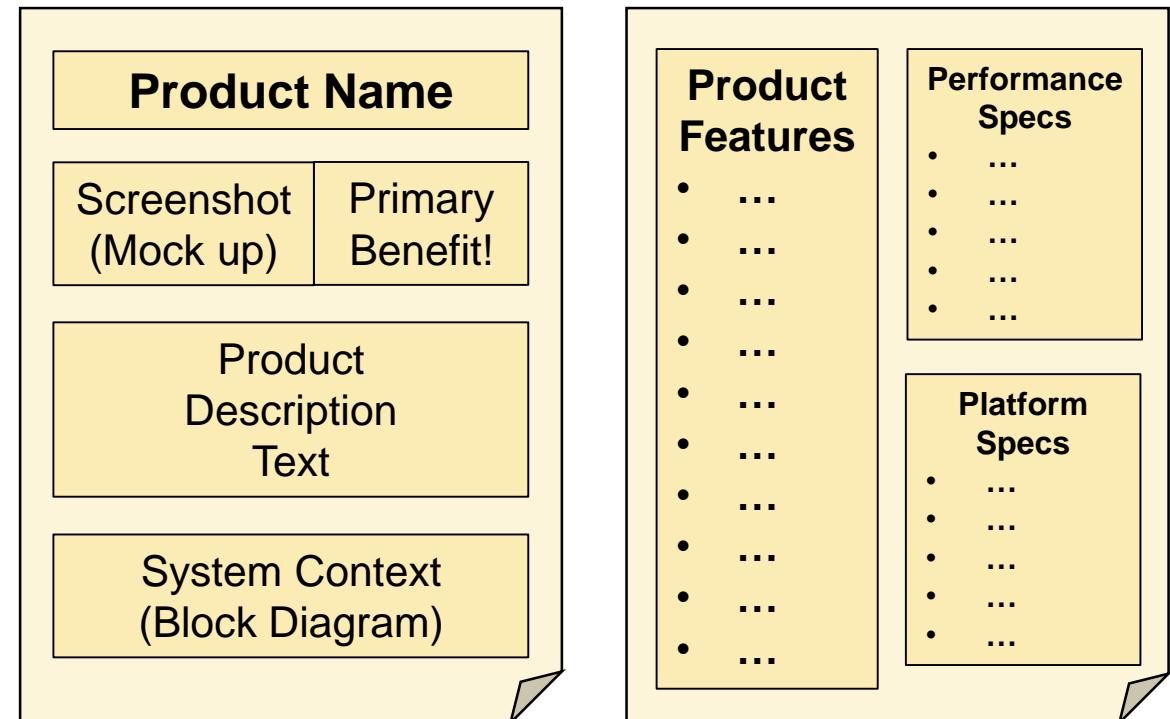
Defining and Communicating the Project Vision: Suitable Documents

a) Vision Document (\rightarrow HBV501G, Lecture 2)

- RUP Vision Document works well at the start of agile projects too (customize as needed)
- Encourages consideration of most relevant business and scope considerations

b) Product Data Sheet

- A two-page “flyer” highlighting product’s key characteristics
- Not just marketing-speak – describe product concisely and precisely
- Not intended for actual publication, but to encourage team to frame vision of final product



Defining and Communicating the Project Vision: Suitable Documents

c) Hypothetical Press Release

- Drafting hypothetical press release for the future release day can help teams to think the project through in user's and management's terms, and clearly articulate its benefits.
- Need to tell complex story in simple, concise way – opportunity to involve marketing dept. and entice internal stakeholders

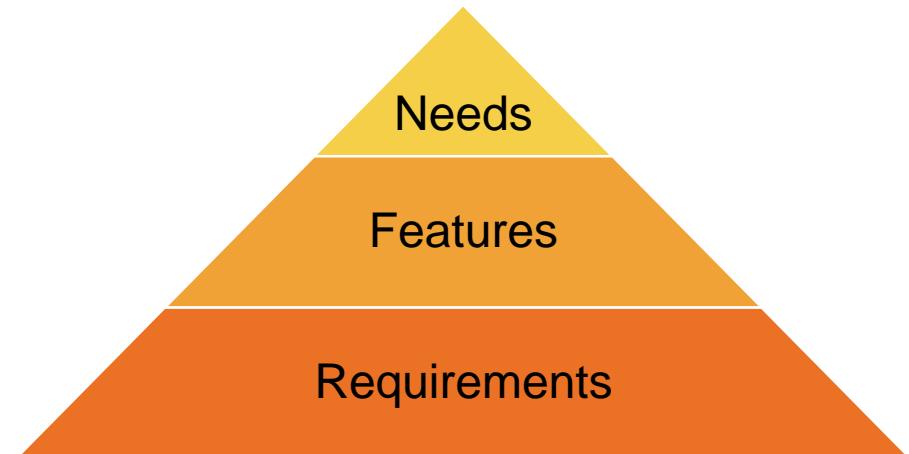
d) Feature Backlog with Briefing

- *If product managers or other responsible stakeholders have a clear enough idea of the project to begin elaborating a preliminary backlog, this can serve as communication basis*
- Have dedicated vision workshop where preliminary backlog is introduced to team, together with discussion of the project motivation
- Preliminary backlog can serve as starting point for effort estimation and prioritization



Needs, Features and Requirements

- The project vision (in whichever form) focuses on **features**
 - i.e. capabilities provided by the system that fulfill one or more stakeholder needs
 - Described in concise key phrases, e.g. “spell checking as you type”
- Features bridge gap between the values that the system shall support (**needs**) and the exact form that this support shall take (**software requirements**)
 - Guideline: Any system (no matter how complex) can be described by < 15-25 key features
 - If you think you need more features, you’re thinking about them on a too detailed level
- Guideline for moving between abstraction levels:
 - First think and talk about **features** as an intuitive starting point for a product scope discussion
 - Then think and talk about **needs** to broaden scope and make sure you didn’t miss anything
 - Then think and talk about **requirements** to go into more detail about how features will be supported



Epics

- Another (optional!) abstraction level on top of features
 - In large projects, a common theme for several features (e.g. “Personalization”)
 - In large organizations, a driver for a particular IT project (e.g. “Move CRM to cloud”)
 - Use only if you feel you need to group your features into larger categories

Differences between abstraction levels:

| | Epic | Feature | Requirement |
|------------------|--|--|---|
| Scope | Strategic direction, overarching business need | Value-delivering product capability supporting an epic | Concrete, functional materialization of part of a feature |
| Origin | Management, Enterprise Architects | Product Owner | Product Owner & Team |
| Timeframe | One or more releases | One or more iterations | One iteration |
| Format | Key phrase | Key phrase or user story | User story or use case |
| Location | Project vision | Project vision | Backlog |

Example Epic, Needs and Features

- **Epic:**
 - In addition to physical store, sell goods also in an online shop.
- **Needs:**
 - Payment in the online shop needs to be effortless, so customers have no reason to change their mind and bail out of a purchase at the last second
 - Customers need to trust the shop so they will return to it to buy more.
 - The shopkeeper needs to minimize their risk of being defrauded by customers in order to remain profitable.
 - [...]
- **Features regarding the need for “effortless payment”:**
 - Payment by credit card
 - Payment by PayPal
 - Payment by invoice
 - Payment by voucher
- **Features regarding the need for “trust in the shop”:**
 - Notification in case of shipping delays
 - Refund for returned goods
 - Secure transmission of payment data
 - Clear display of prices, terms and conditions



Example Requirements

▪ Requirements for feature “Payment by credit card”:

- As a customer,
 - I can pay by credit card for my purchase so I don't need to remember making a payment before/after receiving the goods.
 - I can let the shop store my credit card information so I can pay more conveniently for subsequent purchases.
 - I can delete stored credit card information from the shop's records and thus remain in control of my personal data.
 - I am informed about the success or failure of each credit card transaction so I know if my payment went through.
- As a shopkeeper,
 - I want to accept payments by credit card so I can serve domestic and international customers using the same processes.
 - I can recognize invalid or blocked credit card numbers immediately so I don't need to spend fees on transactions that are bound to fail.
 - I want to receive confirmation of payment before shipping any goods, in order to reduce the risk of fraud.
 - I can reverse credit card payments in case of returns in order to keep my accounting simple.



Self-Check Quiz: Types of Requirements

HBV401G,
Lecture 5

- Indicate if the following are (E)pics, (F)eatures, (N)eeds, (R)equirements or (T)asks:

- a) As a customer, I can let the shop store my credit card information so I can pay more conveniently for subsequent purchases.
- b) Define acceptance test
- c) In addition to physical store, sell goods also in an online shop.
- d) Payment by credit card
- e) Payment in the online shop needs to be effortless, so customers have no reason to change their mind and bail out of a purchase at the last second
- f) Cancel an order



Self-Check Quiz: Types of Requirements (Solution)



- Indicate if the following are (E)pics, (F)eatures, (N)eeds, (R)equirements or (T)asks:

- a) As a customer, I can let the shop store my credit card information so I can pay more conveniently for subsequent purchases. (R)
- b) Define acceptance test (T)
- c) In addition to physical store, sell goods also in an online shop. (E)
- d) Payment by credit card (F)
- e) Payment in the online shop needs to be effortless, so customers have no reason to change their mind and bail out of a purchase at the last second (N)
- f) Cancel an order (F)

User Stories

- In agile projects, requirements are typically expressed as user stories:
 - “As a [role], I can [activity] so that [business value].”
- **User stories serve as a common communication format**
 - short, easy to read (\rightarrow HBV401G, Lecture 3)
 - not precise requirements specifications, but expressions of interest
 - understandable by and valuable to customers
 - understandable by and testable by developers
 - organized in lists that can be flexibly rearranged, rather than large documents
- Don’t get too fixated on the above format though
 - It should encourage you to think about target groups, requirements and rationales
 - But if it gets cumbersome to express sth. this way, write it whichever way yields most clarity
 - e.g. “log in to my customer portal”, “see my daily data usage”, “check my phone bill”

User Stories

- In agile projects, requirements are typically expressed as user stories:
 - “As a [role], I can [activity] so that [business value].”
- **User stories represent “one unit of functionality”**
 - for all purposes of planning and measuring progress (\rightarrow HBV401G, Lectures 4 & 5)
 - small enough that the team can build about half a dozen in an iteration
 - small enough that one can be implemented in a period of days to weeks
 - Not detailed at the outset of the project but elaborated just-in-time
 - This means: You clarify them with the client *before* you start estimating and coding them.
 - This does not mean: You interpret them your way and then ask the customer if you guessed right.
- Besides “requirements” stories, a backlog can also contain other work items
 - e.g. refactoring, bug fixes, support, maintenance, tooling, infrastructure work
 - not phrased like a user story, but sized and treated as such for planning purposes

Refining User Stories

- Formats for expressing epics/needs, features, requirements are somewhat fluid
 - Prefer key phrases for the more high-level abstraction layers
 - Prefer user stories (or even use cases) for the lower abstraction layers
 - Most importantly: Write precisely and appropriately for the given abstraction layer
- Initial versions of user stories may be too broad
 - Some may actually not describe requirements, but features or even epics
 - Rephrase them as such, and consider what requirements would address that feature/epic.
 - Helps to structure requirements, and identify groups of requirements that are not sufficiently explored yet
 - Some may describe requirements too broadly
 - Split them up into several, more precise user stories
 - Helps to refine requirements and obtain more clearly defined, independently implementable functionalities
 - Stepwise refinement of user stories is natural, and a good thing
 - Great opportunity for focused discussion with customer and learning about the business domain



Patterns for Splitting User Stories

| Initial user story | Split user stories |
|---|--|
| As a phone company, I want to update and publish pricing programs to my customer. | Split workflow steps ...I can send a message to the customer's phone. ...I can show a message on the customer web portal. ...I can send an e-mail to the customer's account. |
| | Make business rule variations explicit ...sort by ZIP code. ...sort by home demographics. ...sort by data usage. |
| As a user, I want to be able to select/change my pricing program through my web portal. | Split according to major effort ...I want to use time-of-use pricing. ...I want to prepay for my data and voice usage. ...I want to enroll in critical-peak pricing. |

First story includes
infrastructural groundwork,
others are simpler additions.



Patterns for Splitting User Stories

| Initial user story | Split user stories |
|--|--|
| Focus on the simplest version that can work on its own | |
| As I user, I basically want a fixed price, but I also want to be notified of critical-peak pricing events. | ...I want to be notified of critical peak pricing events. ...I want to be able to switch my plan temporarily. |
| Make procedural variations explicit | |
| As a phone company, I can send messages to customers. | ...regarding their contract. ...regarding promotional offers. ...regarding general news. |
| Defer system qualities | |
| As a user, I can view my data usage in various graphs. | ...using bar charts that compare weekly usage. ...in a comparison chart, so I can compare my usage to people from similar demographics. |

Build a simple version first, and leave more complex story for later implementation.



Patterns for Splitting User Stories

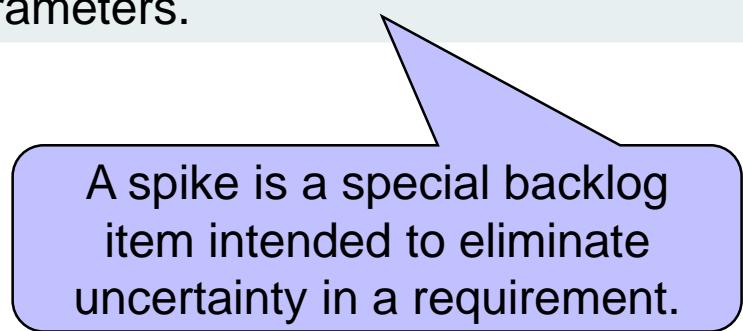
| Initial user story | Split user stories |
|---|---|
| | Split operations As a user, I can manage my account. ...I can sign up for an account. ...I can edit my account settings. ...I can cancel my account. ...I can add more devices to my account. |
| I want to transfer credits from the loyalty program of partner company X. | Use case scenarios <i>Main scenario:</i> Notify companies of request and handle transfer: [...] <i>Alternative scenario:</i> Handle exceptions in the transfer process: [...] |

Complex interactions may be more precisely and efficiently described as use cases.



Splitting User Stories

| Initial user story | Split user stories |
|--|---|
| Research more to eliminate uncertainty (“spike”) | |
| As a consumer, I want to see my real-time and historic data usage visualized so that I can intuitively adjust quality-of-service parameters as needed. | <p>Technical spike: Research what level of real-time usage measuring and display is feasible, what degrees of QoS the company is willing to support, and how soon any user adjustments can take effect.</p> <p>Functional spike: Prototype a histogram in the web portal and get some user feedback on chart style and adjustment parameters.</p> |



A spike is a special backlog item intended to eliminate uncertainty in a requirement.



Dealing With Uncertainty in Spikes

- “**Spikes**” are special backlog items aimed at reducing risk and uncertainty, e.g.
 - Performing basic research, familiarizing team with new technology or business domain
 - Analyzing a complex user story to see how it can be broken down into estimable pieces
 - Prototyping to gain confidence in a technology and make more informed estimate
- **Technical spikes** aim to research technical approaches, e.g.
 - Evaluate implementation technologies, make build-or-by decisions
 - Evaluate a user story’s performance or load impact
 - Gain confidence with an approach before committing to an estimate
- **Functional spikes** aim to understand how users will work with the system, e.g.
 - Prototype interface through mock-ups, wireframes, storyboards etc.
 - Get feedback from stakeholders



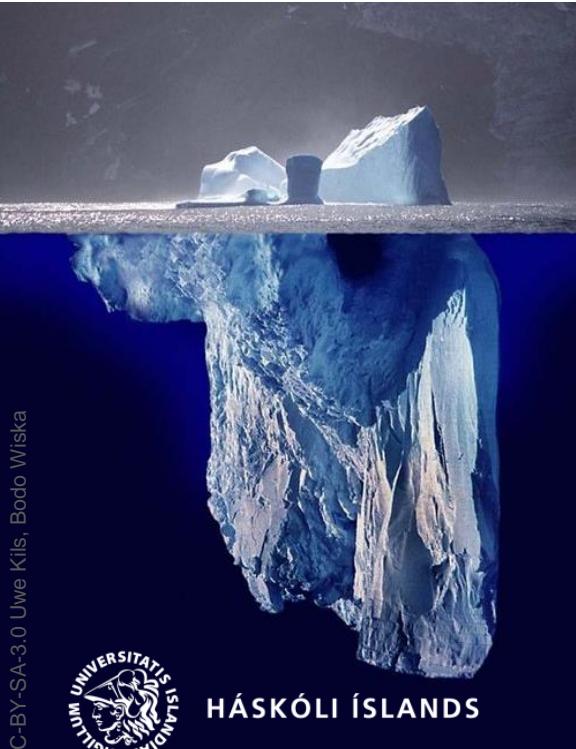
Dealing With Uncertainty in Spikes

- Spikes do not directly deliver *user value*
- Use sparingly and with caution!
- Guidelines:
 - Spikes are treated like other backlog items, i.e. estimated and sized to fit in an iteration
 - Spikes do not produce working code, but information, e.g. a prototype or basis for decision
 - Spike results should still be concrete and demonstrable (helps to justify effort to stakeholders)
 - Spikes should be the exception, not the rule
 - Each user story has inherent uncertainty, and will therefore involve some spike-like research
 - Explicit spike backlog items should be reserved for larger, more critical uncertainties
 - Spike should not be planned in the same iteration as the user stories relying on its outcome
 - Otherwise, the planning of the user story cannot benefit from the knowledge gained in the spike



User Stories Are Just the Tip of the Requirements Iceberg

- Even after refining user stories, you may find they still feel quite open-ended.
 - This is by design, since agile approaches deliberately eschew detailed requirements specs.
- You still need a more precise idea of user stories' intention to implement them.
 - The required knowledge can take many shapes:

- 
- **User story**
 - visible anchor of customer expectation, team commitment and requirements knowledge
 - Knowledge from conversations with stakeholders
 - Attachments (e.g. examples, models, prototypes, use cases, data etc.)
 - Acceptance criteria (conditions of satisfaction placed on system)
 - Tasks (steps to be taken to ensure that acceptance criteria are satisfied)

Acceptance Criteria

- Acceptance criteria are conditions of satisfaction being placed on the system.
 - “Based on which criteria can the customer and supplier agree that this user story has been implemented satisfactorily?”
- Acceptance criteria are not functional or unit tests, but are a first step towards making user stories evaluable against concrete criteria.
 - In effect, they are yet one more detailed level of the user story’s definition
 - Describing the solution in business terms
- Example user story:
 - As a consumer, I want to be able to see my daily energy usage so that I can lower my consumption and cost.
- Example acceptance criteria:
 - Read deciwatt meter data every 10 seconds and display on portal in 15-minute increments.
 - Read kilowatt meter for new data as available and display on the portal every hour.

Issues with *User Stories*

- User stories don't give developers a context to work from
 - When is the user doing this? What is the context of the operation? What is their larger goal?
- User stories don't give any sense of scope or completeness
 - How "deep" is a user story? What exceptions lurk inside? What's outside these spotlights?
- User stories don't provide a mechanism for looking ahead at upcoming work
 - Hard to foresee main and alternate scenarios; extensions usually recognized mid-iteration
- A list of backlog items is easy to comprehend, manage and prioritize, but inadequate to provide the context required to design larger systems.



Benefits of Use Cases (Yes, in Agile Projects!)

- A UC's main success scenario states exactly what the system will and will not do
 - Provides context for each requirement, and agreement on scope between biz and tech side
- A UC's alternate scenarios answer the many tricky, detailed business questions that no business person initially thinks of but every developer soon runs into:
 - What are we supposed to do in this special case? What if that goes wrong in the process?
- A UC's list of goals provides a summary of the system's benefits & added value
 - Helpful for initial prioritizing, team allocation and scheduling
- A UC's extension conditions highlight business and technical complexity drivers
 - These would otherwise surface while a requirement is worked on, and delay it unforeseeably
- Full set of UCs shows that developers have considered each user's needs, each business goal and variant.
 - Gives customer and team more confidence that estimates and schedules are realistic.



Capturing Use Cases

1. Identify and describe the actors

- Who uses the system?
- Who gets information from this system?
- Who provides information to the system?
- Where is the system used?
- Who supports and maintains the system?
- What other systems or devices use this system?

2. Identify the use cases

- What will the actor use the system for?
- Will the actor create, store, change, remove, or read data in the system?
- Will the actor need to inform the system about external events or changes?
- Will the actor need to be informed about certain occurrences in the system?



Capturing Use Cases

3. Identify the actor and use case relationships

- Which actor initiates the use case?
- Which actors are participating in the use case?

4. Outline the flow of the use cases

- Basic flow (main success scenario)
 - What actor's event starts the use case?
 - How does the use case end?
 - How does the use case repeat some behavior?
- Alternate flows (alternate scenarios)
 - What else can the actor do?
 - How will the actor react to optional situations?
 - What variants might happen?
 - What exceptions to the usual behavior may occur?



Capturing Use Cases

5. Refine the use cases

- What are alternate flows, including unusual exception conditions?
- What preconditions must be satisfied before a use case can start?
- What exit conditions will the use case leave behind?
 - Success guarantee (state after successful execution)
 - Minimum guarantee (state after unsuccessful execution)



Use Cases in an Agile Context

- Use cases can be useful “inside” and “outside” user stories
 - “**inside**”: A use case is attached to a particular user story, in order to specify how exactly the interaction demanded by the story shall play out
 - “**outside**”: A use case describes a complex interactive feature, whose realization will be broken into several user stories for project planning purposes
- Guidelines: When applying use cases in agile projects,
 - Keep them lightweight, i.e. no design details, GUI specs etc.
 - Don’t treat them like fixed requirements
 - Like user stories, they just state intended system behavior
 - Model them informally, and don’t worry too much about maintaining them
 - They are tools to help you understand the requirements, not to document them
- Use cases are a proven tool to think through complex system requirements
 - Highly recommended to use them independently of the chosen software process model

Tasks

- Once a backlog item (user story or other work item) is sufficiently understood in terms of the *requirements* it poses...
- ...it's time to consider *what needs to be done* to implement those requirements
 - i.e. which concrete activities team members need to perform to realize the backlog item
 - Def.: A task is a small unit of work that is necessary for the completion of a backlog item.
- This also is a form of refinement of the user story
 - but while *acceptance criteria* refine the user story in *business terms* (product/user view), *tasks* refine it in *technical, operational terms* (project/developer view)
- Tasks help the team to understand what needs to be done
 - most precise basis for estimating efforts
 - interface between analysis/design and construction/controlling work in each iteration



Example Tasks

- **User story:**
 - Select photo for upload
- **Tasks for story:**
 - Define acceptance test
 - Code story
 - Code acceptance test
 - Get test to pass
 - Document feature in user help



Summary: Agile Requirements Engineering

- Start by brainstorming on **features**
- Take a step back and consider **needs**, potentially **epics**
- Take a closer look and consider **requirements** (typ. in user story format)
- Schedule **spikes** to resolve uncertainties
- Refine **user stories**
 - Discuss with customer
 - Elaborate and/or split
 - Work out **use cases** to understand particularly complex scenarios
- Define **acceptance criteria**
- Define **tasks** to be performed in order to satisfy acceptance criteria
- **Prioritize requirements, estimate efforts, plan iterations**



Reflection: Requirements Engineering in Agile vs. Plan-driven Projects

- **Similarities** between agile and plan-driven requirements engineering:
 - Expect to spend just as much time on requirements work in either process
 - This is natural: Impossible to circumvent the need and effort of understanding the business domain well enough to implement the system correctly
 - Requires a lot of learning, and critical reflection of one's own knowledge and assumptions
 - Requires close collaboration with the customer
- The **differences** are that agile requirements engineering...
 - ...sees written requirements as starting points of clarification, rather than endpoints
 - ...permeates the whole project, rather than occurring in explicitly planned phases
 - ...takes many forms (user stories, discussions), rather than materializing in one document
- **Refactoring is not an efficient alternative to diligent requirements work!**
 - Refactor mostly to integrate new features into existing “minimum viable” code (pragmatic)...
 - ...but don't keep refactoring things just because you didn't think them through (expensive)

Recap: Assignment 1: Project Plan and Requirements

- By **Sun 2 Feb**, submit a **project outline** in Ugla, containing:

1. Project vision (in form of a RUP Vision document, Data Sheet or Press Release)
2. Product backlog (prioritized user stories)
3. User Story estimates (three-point estimates of expected cases, using basic PERT formula)
4. Project schedule (dates for sprints, milestones, assignments 2 and 3)

Start on #1&2
this week

- On **Thu 6 Feb**, present and **explain** your project outline to your tutor:
 - What considerations influenced your choice of scope, estimates and schedule?

Start on #3&4
next week

- **Grading criteria:**

- Project vision is clear and plausible (15% of this assignment's grade)
- Product backlog describes requirements clearly and with realistic scope (40%)
- Project estimates calculated using appropriate formulae (35%)
- Project schedule is clear, realistic, and specifies dates for assignments 2 and 3 (10%)





Hugbúnaðarverkefni 2 / Software Project 2

3. Software Estimation

HBV601G – Spring 2020

Matthias Book



HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRVÍSINDASVIÐ
ÍÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD

In-Class Quiz Prep

- Please prepare a small scrap of paper with the following information:

ID: _____ @hi.is Date: _____

1. a) _____ b) _____

2. a) _____ - _____ b) _____ - _____

c) _____ - _____ d) _____ - _____

e) _____ - _____

- During class, I'll show you questions that you can answer very briefly
 - Just numbers or letters, no elaboration
- Hand in your scrap at the end of class
- All questions in a quiz weigh same
- All quizzes (ca. 10-12 throughout semester) have the same weight
 - Your worst 2 quizzes will be disregarded
- Overall quiz grade counts as optional question worth 7.5% on final exam



Software Estimation

see also:

- McConnell: Software Estimation, Ch. 1, 3-5, 7-13
- Leffingwell: Agile Software Requirements, Ch. 8



Accurate Estimates Are Based on Precise Requirements

A seemingly trivial requirement:

Users must enter a valid phone number with their order.

Each of these can introduce a factor 10 of difference in complexity, quality, time etc.

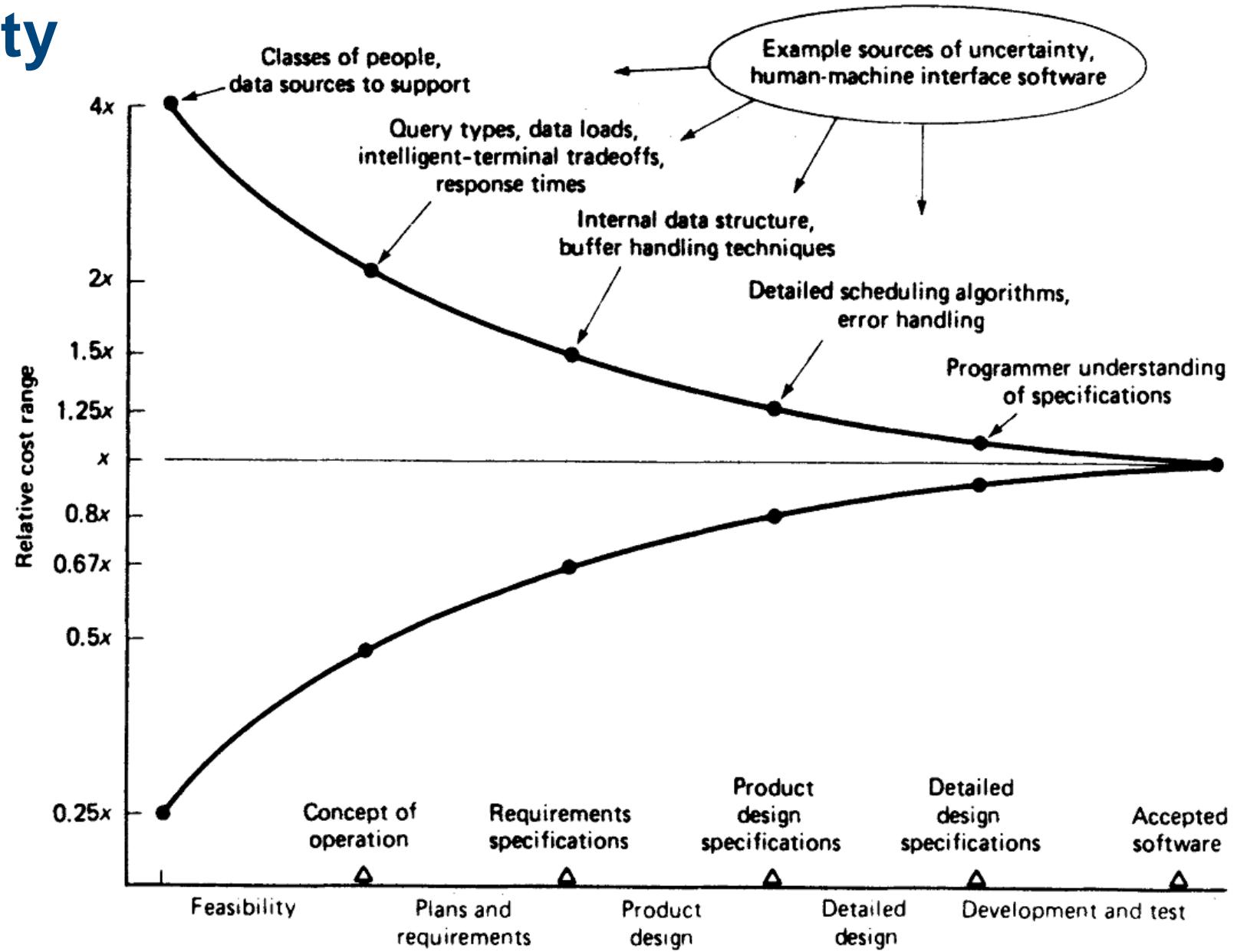
But we need to understand:

- This is not just about *entering*, but about *validating* a number using some “phone number checker (PNC)”
- If the customer wants a PNC, will he want the version we can build in 2 hours, 2 days, or 2 weeks?
- If we implement the cheap version of the PNC, will the customer later want the expensive version after all?
- Can we use an off-the-shelf PNC, or are there design constraints that require us to develop our own?
- Do the PNC and the Address Checker need to interact? How long will it take to integrate them?
- How will the PNC be designed?
- How long will it take to code the PNC?
- What will the quality level of the PNC be?
- How long will it take to debug and correct mistakes made in the PNC implementation?



Cone of Uncertainty

- A measure for the maximum possible estimation accuracy over the course of a project
 - Your cone may be wider
 - i.e. your accuracy may be worse!
- The cone does not narrow by itself
 - Continuous active work is required to eliminate sources of uncertainty in the project.

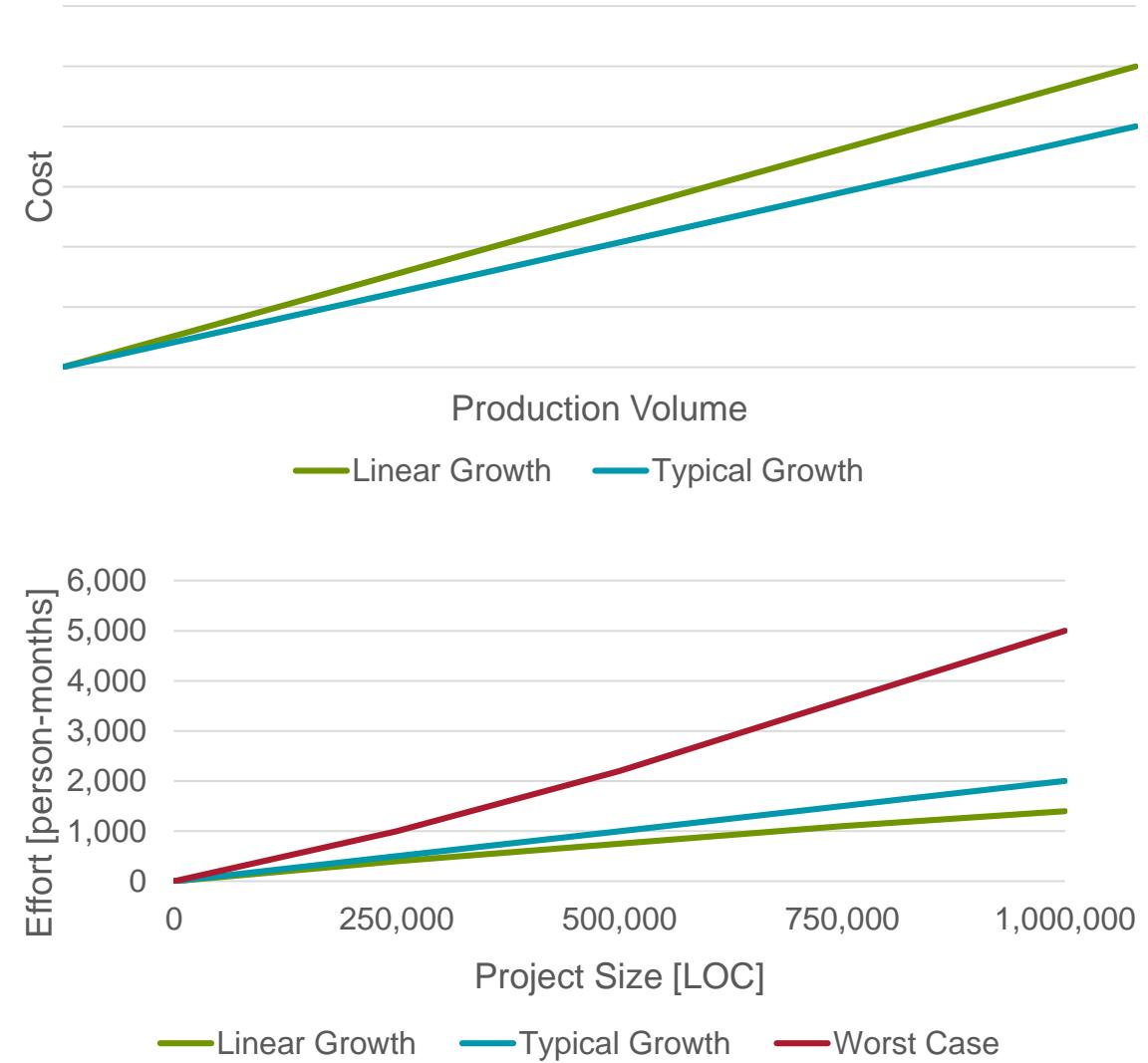


Impact Bandwidth of Factors Affecting Project Effort



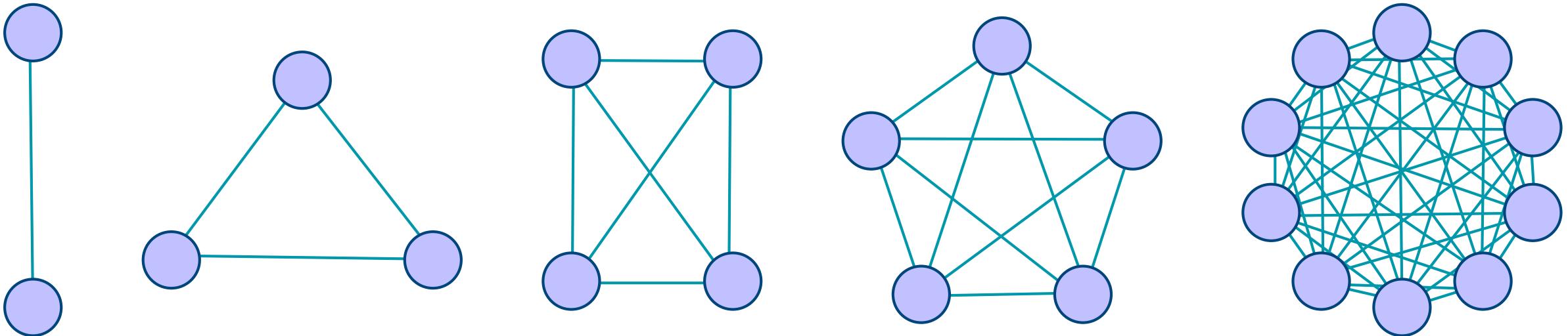
Diseconomies of Scale

- Many manufacturing domains can benefit from **economies of scale**:
“The more units we produce, the smaller the cost of each unit.”
- In software engineering, we have to deal with a **diseconomy of scale**:
“The larger a system we build, the higher the cost of building each part.”
- Effort does not scale linearly with project size, but exponentially.



Diseconomies of Scale

- A system that is 10 times as large requires *more* than 10 times as much effort.
 - A linear increase in components (or a linear increase in team size) results in an exponential increase in **relationships** between components (or people).
 - Of course, efficient component design and encapsulation (or effective team structures) can reduce the number of actual relationships considerably,
 - but the cognitive complexity that the team has to deal with remains high.



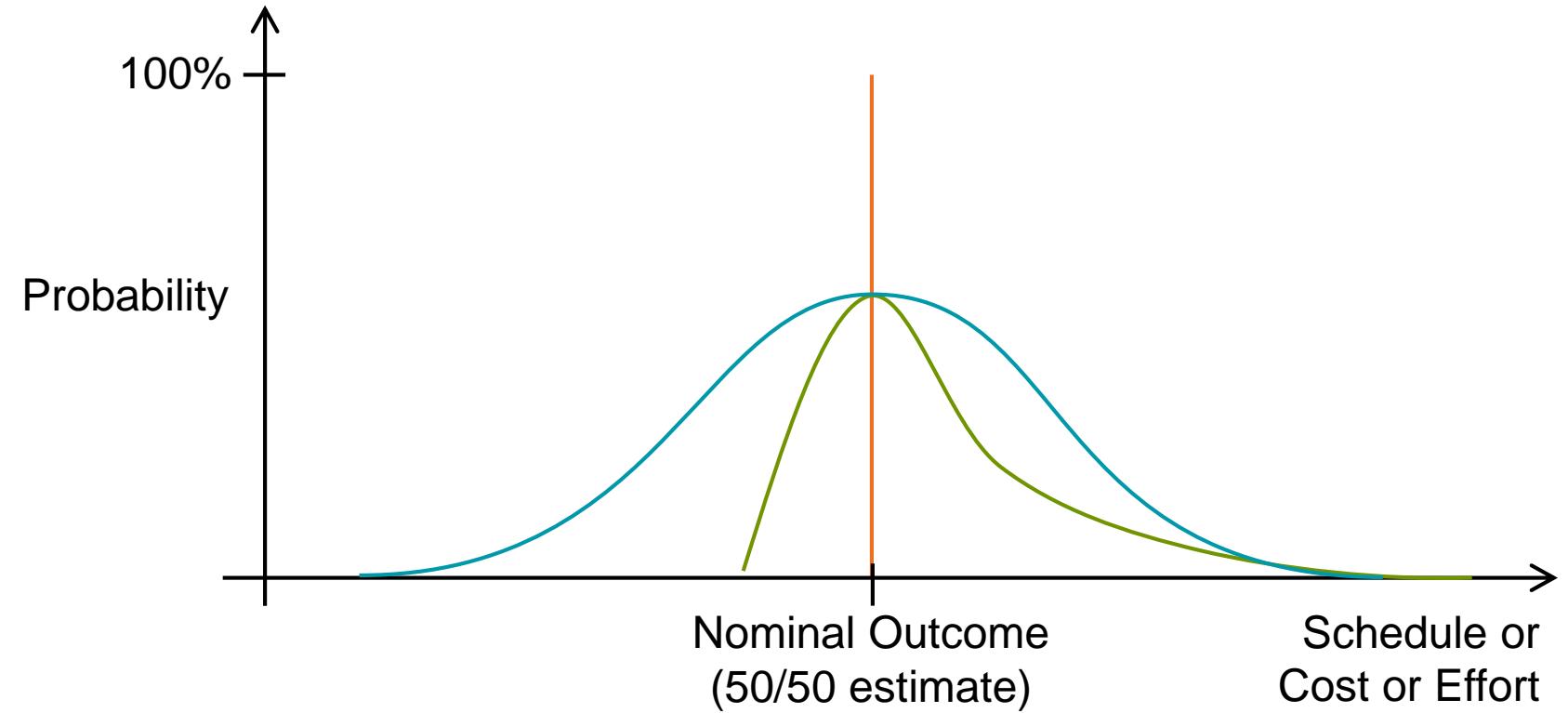
Difficulties in Software Estimation

- **Caution when interpreting management requests**
 - Determine whether you are expected to estimate, or figure out how to hit a target
- **Caution when preparing estimates**
 - Hard to quantify the subject
 - Lots of confounding factors
 - Temptation to tweak the estimate
- **Caution when letting management interpret your estimates**
 - Your estimates may be understood as precise commitments



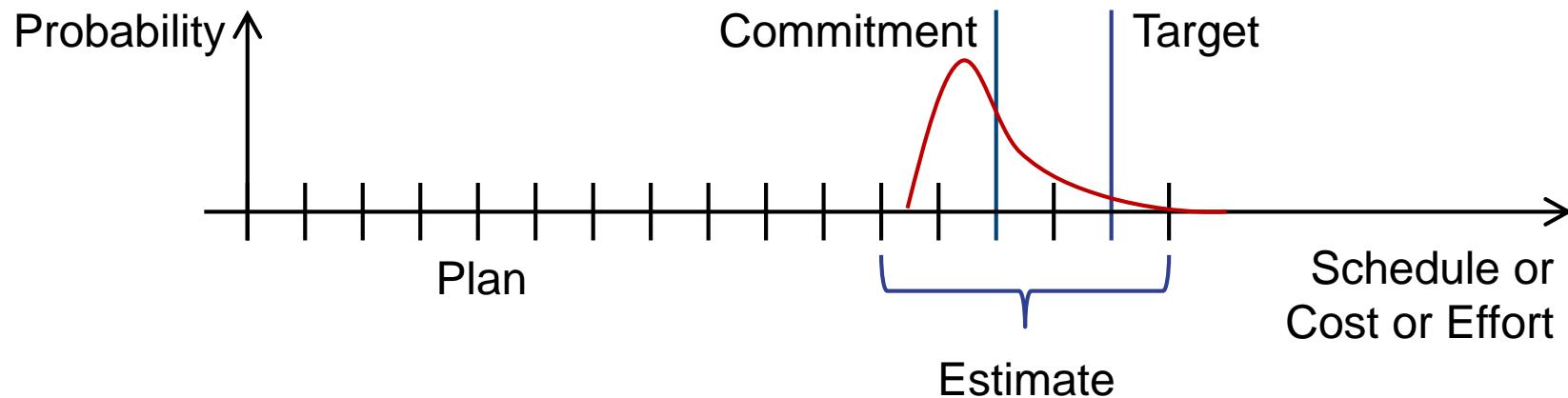
Estimates as Probability Statements

- Single point estimate: Assumes that estimate will accurately predict result
- Bell curve: Disregards limits of how efficiently a project team can complete work
- ✓ Realistic curve: There's a limit to how *well* a project can go, but not how *bad*



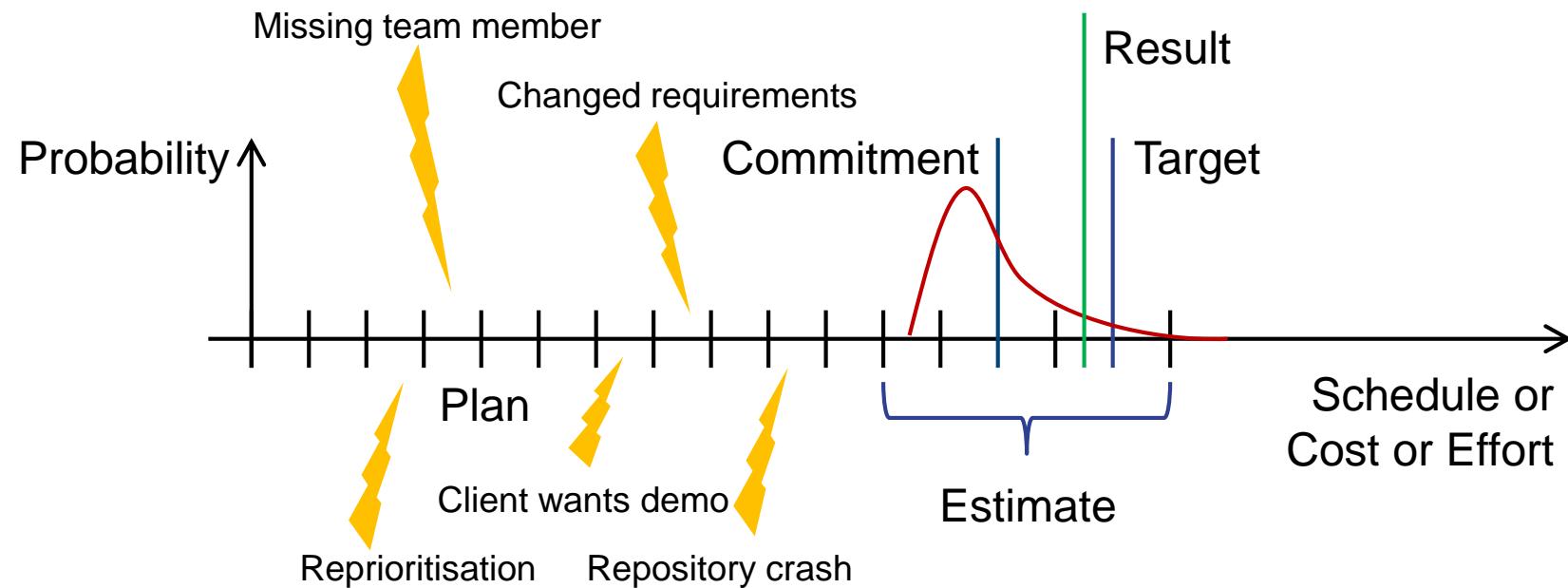
Terminology

- **Target:** fixed value (deadline, budget etc.) determined by external factors
- **Estimate:** preliminary prediction of a future value/interval
- **Commitment:** Intention of reaching a certain future value
- **Plan:** Agreement of steps to achieve the commitment



Life Is What Happens While You Are Making Other Plans

- Some project events make (the foundations of) prior estimates obsolete.



The Purpose of Software Estimation

- The aim of effort estimates is not to predict the project result, but to **judge whether the project target is sufficiently realistic to be achievable through corresponding project management.**
 - Estimates do not need to be extremely precise, as long as they are **useful**.
- Experience: If the initial target and the initial estimate are within ~20% of each other, the project manager will have enough maneuvering room (in terms of features, schedule, staffing etc.) to meet the project's business goals.
- If the gap between target and estimate is larger, it is very unlikely the project can be completed successfully.
 - Bring target into better alignment with reality before the project has a chance of success.
 - **RESIST TEMPTATION / PRESSURE** to align estimate better with target!
 - Your first estimate is rough but honest.
 - Revisions not based on additional data are just kidding yourself and endangering your project.

Estimation Approaches Discussed in This Class

A. Projection from counts

1. Concrete items
2. Function points

B. Individual expert judgment

C. Group expert judgment

1. Wideband Delphi
2. Planning Poker

D. Qualitative estimation



A₁) Projection from Counts

1. Find a suitable type of item to count

- Highly correlated with size of software you're estimating
- Available early in development cycle
- Statistically meaningful number of items available (>20)
 - If we are talking about less, the individual differences between these items become too dominant
- Countable with minimal effort
- Historical averages based on comparable parameters(!) available
 - Similar product, similar team, similar technology, similar complexity...

2. Count items of interest

3. Calculate effort from item counts, based on average effort per item



Countable Items for Projecting Estimates

- Marketing requirements
- Engineering requirements
- Features
- Use cases
- User stories
- Function points
- Change requests
- Web pages
- Reports
- Dialog boxes
- Database tables
- Classes
- Defects found
- Configuration settings
- Lines of code already written
- Test cases already written
- ...
- **Projection:** Multiply counted items with average effort per item observed in historical projects of similar kind



In-Class Quiz #1: Projecting Estimates from Counts



a) Counting defects late in project

- For the last 250 defects we fixed, we needed 2 hours per defect on average.
- 400 defects are currently open in our project.
- How many hours will it approximately take us to fix the open defects?

b) Counting web pages

- So far, each dynamic web page in the project cost us a total of 40 hours to design, code and test, on average.
- We have 10 web pages left to build.
- How many hours will it approximately take us to finish this?

A₂) Function Points (FP)

- Synthetic measure of software size assigned to counts of various types of items:
- **External inputs**
 - Screens, forms, dialog boxes, control signals through which an end user or other system adds, deletes, or changes our system's data
- **External outputs**
 - Screens, reports, graphs, control signals that our system generates for use by an end user or other system.
- **Internal logical files**
 - Major logical groups of end-user data or control information completely controlled by our system, e.g. a single flat file or a single table in a database
- **External interface files**
 - Files controlled by other systems with which our system interacts; this includes each major logical group of data or control information entering or leaving our system



Function Points

- Each system characteristic is classified by complexity (low / medium / high) and assigned an according number of function points:

| System Characteristic | Low Complexity | Medium Complexity | High Complexity |
|--------------------------|----------------|-------------------|-----------------|
| External inputs | 3 | 4 | 6 |
| External outputs | 4 | 5 | 7 |
| Internal logical files | 4 | 10 | 15 |
| External interface files | 5 | 7 | 10 |

- FPs are an abstract, relative measure of complexity, not absolute size/time/effort
- Formulas exist to translate FPs into lines of code, based on industry experience
 - e.g. in Java, 1 FP requires at least 40, typically 55, at most 80 lines of code
- More useful: Use organizational experience to translate FP to expected effort



Recommendation: Avoid in practice – use story points if you want an abstract metric

Lines of Code (LOC)

Traditional size measure found in many tools and textbooks

Advantages

- Data easily collected and evaluated by tools
- Lots of historical data exists
- Effort per line of code is roughly constant across languages
 - (*Effect* per line of code obviously is not!)
- Most convertible “currency” between projects
 - But make sure projects are comparable to draw meaningful conclusions!

Disadvantages

- LOC don't reflect
 - Diseconomies of scale in software development effort
 - Difference in programmer productivity
 - Difference in expressiveness of programming languages
- Using LOC for estimating non-coding work (requirements, design etc.) is counterintuitive
- Which Lines of Code to count?

Recommendation: Avoid in practice – mostly useful for academic purposes



Experiment: Estimation Confidence

Estimate a range in which you are 90% confident the actual value will be:

- a) Latitude of San Francisco: ■ _____ - _____ °N
- b) Birth year of Fridtjof Nansen: ■ _____ - _____
- c) Area of Iceland: ■ _____ - _____ km²
- d) Production cost of movie “Avatar”: ■ _____ - _____ million US\$
- e) Registered students at HÍ: ■ _____ - _____



Experiment: Estimation Confidence

How many of the following actual values fall inside your estimation range?

a) Latitude of San Francisco:

- 37°47' N

b) Birth year of Fridtjof Nansen:

- 1861

c) Area of Iceland:

- 103.000 km²

d) Production cost of movie “Avatar”:

- 237 million US\$

e) Registered students at HÍ:

- 13.092 (as of 20 Oct 2019)

| In range | Percentage |
|----------|------------|
| 0 of 5 | 0% |
| 1 of 5 | 20% |
| 2 of 5 | 40% |
| 3 of 5 | 60% |
| 4 of 5 | 80% |
| 5 of 5 | 100% |



Lesson: Estimation Accuracy

- Remember you were asked to provide a range that you were *90% confident* to include the actual value.
- Reflect:
 - Did you subconsciously try to make the estimate “better” by tightening the range?
 - Were you still highly confident that your range included the actual value?
- Perceived estimation confidence is usually much higher than actual confidence
 - Be cautious with statements like “90% confidence” – usually your estimate is far less reliable
 - Choose your estimation range deliberately
 - Don’t make the range narrower than necessary
 - The estimation range should reflect your inconfidence



Estimation vs. Commitment

- In software projects, people (and possibly yourself) will assume that your estimates can be directly translated into commitments.



Estimation vs. Commitment

- Your estimate is much less reliable than you think, but other people even think it's much more reliable!
 - Carefully manage expectations and be clear about your confidence



B) Individual Expert Judgment

- Let an expert (typically: developer) come up with an estimate for each feature
 - Caution: subjective, possibly based on non-transferable experience / incorrect assumptions
- Avoid developer single-point estimates
 - Usually, these reflect the developer's subconscious *best case* assumption!
- Create three-point estimates: Best Case, Most Likely Case, Worst Case
 - This will stimulate developers to think about full range of possible outcomes for each feature
- Calculate expected case from three-point estimates for each feature:
 - $\text{ExpectedCase} = (\text{BestCase} + 4 * \text{MostLikelyCase} + \text{WorstCase}) / 6$ (basic PERT formula)
 - $\text{ExpectedCase} = (\text{BestCase} + 3 * \text{MostLikelyCase} + 2 * \text{WorstCase}) / 6$
 - (alternative formula if you feel your team's "most likely" estimates tend to be too optimistic)

Example: Calculating Expected Cases

Weighted average, e.g.
 $\text{ExpectedCase} = (\text{BestCase} + 4 * \text{MostLikelyCase} + \text{WorstCase}) / 6$

| Feature | Best Case | Most Likely Case | Worst Case | Expected Case |
|--------------|-----------|------------------|------------|---------------|
| 1 | 1.25 | 1.5 | 2.0 | 1.54 |
| 2 | 1.5 | 1.75 | 2.5 | 1.83 |
| 3 | 2.0 | 2.25 | 3.0 | 2.33 |
| 4 | 0.75 | 1 | 2.0 | 1.13 |
| 5 | 0.5 | 0.75 | 1.25 | 0.79 |
| 6 | 0.25 | 0.5 | 0.5 | 0.46 |
| 7 | 1.5 | 2 | 2.5 | 2.00 |
| 8 | 1.0 | 1.25 | 1.5 | 1.25 |
| 9 | 0.5 | 0.75 | 1.0 | 0.75 |
| 10 | 1.25 | 1.5 | 2.0 | 1.54 |
| Total | | | | 13.62 |

[Days to Complete]



Checklist for Individual Estimates

- Is what's being estimated clearly defined?
- Does the estimate include all the kinds of work needed to complete the task?
- Does the estimate include all the functionality areas needed to complete the task?
- Is the estimate broken down into enough detail to expose hidden work?
- Did you look at documented facts from past work or estimate purely from memory?
- Is the estimate approved by the person who will actually do the work?
- Is the assumed productivity similar to what has been achieved on similar assignments?
- Does the estimate include a Best Case, Worst Case, and Most Likely Case?
- Is the Worst Case really the worst case? Does it need to be made even worse?
- Is the Expected Case computed appropriately from other cases?
- Are you aware of the assumptions that are underlying this estimate?
- Has the situation changed since the estimate was prepared?

Training Your Estimation Accuracy Over Time

To improve estimation accuracy:

- Compare actual results to estimated results
 - Calculate Magnitude of Relative Error (MRE) for each feature estimate:
$$\text{MRE} = |(\text{ActualResult} - \text{EstimatedResult}) / \text{ActualResult}|$$
 - The average of your MREs per sprint should decrease over time
 - Check if actual results were in range between best case and worst case
 - The percentage of actual results that are in this range should increase over time
 - Check if 50% of expected cases were overrun and 50% of them were underrun
 - If not, your individual estimates tend to be too optimistic or too pessimistic
 - Work on improving your best, most-likely and worst-case estimation accuracy
- Try to understand
 - What went right and what went wrong
 - What you overlooked
 - How to avoid making those mistakes in the future



Example: Calculating Magnitude of Relative Error

| Feature | Best Case | Most Likely Case | Worst Case | Expected Case | Actual Outcome | MRE | Between best and worst case |
|--------------|-----------|------------------|------------|---------------|----------------|-----------------|-----------------------------|
| 1 | 1.25 | 1.5 | 2.0 | 1.54 | 2 | 23% | Yes |
| 2 | 1.5 | 1.75 | 2.5 | 1.83 | 2.5 | 27% | Yes |
| 3 | 2.0 | 2.25 | 3.0 | 2.33 | 1.25 | 87% | No |
| 4 | 0.75 | 1 | 2.0 | 1.13 | 1.5 | 25% | Yes |
| 5 | 0.5 | 0.75 | 1.25 | 0.79 | 1 | 21% | Yes |
| 6 | 0.25 | 0.5 | 0.5 | 0.46 | 0.5 | 8% | Yes |
| 7 | 1.5 | 2 | 2.5 | 2.00 | 3 | 33% | No |
| 8 | 1.0 | 1.25 | 1.5 | 1.25 | 1.5 | 17% | Yes |
| 9 | 0.5 | 0.75 | 1.0 | 0.75 | 1 | 25% | Yes |
| 10 | 1.25 | 1.5 | 2.0 | 1.54 | 2 | 23% | Yes |
| Total | | | | 13.62 | 16.25 | 29% avg. | 80% yes |





Hugbúnaðarverkefni 2 / Software Project 2

4. Agile Estimation

HBV601G – Spring 2020

Matthias Book



HÁSKÓLI ÍSLANDS
VERKFRAÐI- OG NÁTTÚRVÍSINDASVIÐ
ÍÐNAÐARVERKFRAÐI-, VÉLAVERKFRAÐI-
OG TÖLVUNARFRÆÐIDEILD

In-Class Quiz Prep

- Please prepare a small scrap of paper with the following information:

ID: _____ @hi.is Date: _____

a) _____

d) _____

b) _____

e) _____

c) _____

- During class, I'll show you questions that you can answer very briefly
 - Just numbers or letters, no elaboration
- Hand in your scrap at the end of class
- All questions in a quiz weigh same
- All quizzes (ca. 10-12 throughout semester) have the same weight
 - Your worst 2 quizzes will be disregarded
- Overall quiz grade counts as optional question worth 7.5% on final exam



Update: Team Consultations

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | |
|---|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|---|
| A | | X | | X | | | X | | | X | | X | X | | | | | | | | X | | | | | | X | | X | | |
| K | | | | | | X | | X | X | | X | | | | | | | | | | X | | X | X | X | X | | | X | | |
| S | X | | X | | X | | | | | | | | | X | X | | | | | | | X | | X | X | | X | | | | |
| # | 2 | 4 | 3 | 3 | 4 | 4 | 3 | 4 | 3 | 4 | 4 | 3 | 4 | | 4 | 3 | | | | | 3 | 4 | 3 | 4 | 4 | 3 | 3 | 2 | 2 | 4 | 3 |

▪ Consultation time slots

- Thursdays 13:20-16:30, V-152
 - Teams 1-10: 13:20-14:20
 - Teams 11-20: 14:25-15:25
 - Teams 21-30: 15:30-16:30

▪ Seating arrangement

- Arnar's teams: last rows
- Kristján's teams: middle rows
- Sigurður's teams: first rows

▪ Team/Tutor affiliations rotated

- To enable fresh feedback and avoid grading biases
- Table above reflects new affiliations
- Please let your tutor know if you're not planning to attend a consultation
 - so they don't hang around waiting for you



Update: Assignment 1: Project Plan and Requirements

- By Sun 2 Feb, submit a **project outline** in Uglá, containing:
 1. Project vision (in form of a RUP Vision document, Data Sheet or Press Release)
 - If writing a new vision document: Write Sect. 1.5, another 1.x, another 2.x (except 2.1)
 - If revising your vision document from HBV501G:
 - a) Rewrite your whole existing document substantially to focus on the mobile aspects, OR
 - b) Add another Sect. 1.x, another Sect. 2.x (except 2.1), and another Sect. 3.x
 2. Product backlog (prioritized user stories)
 - Cut HBV501G backlog down to make it realistic, but also include mobile-specific stories if possible
 3. User Story estimates (three-point estimates of expected cases, using basic PERT formula)
 4. Project schedule (dates for sprints, milestones, Ass. 2 **submission** and Ass. 3 **code exchange**)
- **Grading criteria:**
 - Project vision is clear and plausible (15% of this assignment's grade)
 - Product backlog describes requirements clearly and with realistic scope (40%)
 - Project estimates calculated using appropriate formulae (35%)
 - Project schedule is clear, realistic, and specifies dates for assignments 2 and 3 (10%)

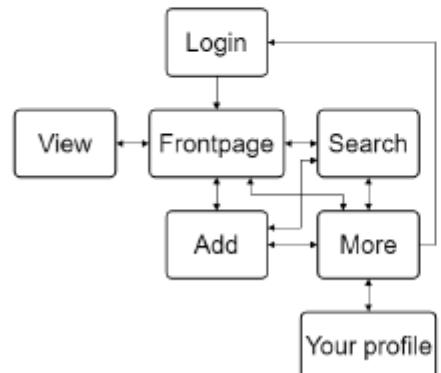


Example: Data Sheet

Eventster



System Context



Primary benefits

- Á Eventster er hægt að skipuleggja viðburði með vinum sínum.
- Halda utan um viðburði sem þú hefur áhuga á.
- Séð hvaða viðburði vinir þínir ætla á.
- Fundið nýja viðburði sem þú vissir ekki af þínum.
- Fundið viðburði tengda áhugamálum.
- Eventster er frábær leið til að eignast nýja vini.

Product Description

- Á Eventster geta notendur búið til hópa og viðburði sýnilegum meðlimum
- Á Eventster er hægt að skrá sig á viðburði og séð hvaða notendur eru skráðir á þeim.
- Allir viðburðir á Eventster eru búnir til af notendum og því verður gríðarlega fjölbreytt úrval.
- Á Eventster getur þú talað við aðra notendur í gegnum commentakerfi.

Product Features

- Sign up and Login to your personal account.
- Edit your user information.
- Add image to your profile, events and groups.
- Create an event/group, both public and private.
- Add other users to your private group.
- View groups, events, users.
- Mark yourself as attending on an event, so that it is visible to all users.
- Search for events/groups/users.
- Filter events by category.
- Comment on event.
- Reply to comment.

Platform Specs

- Appið er keyranlegt í sínum með android stýrikerfi.
- Ekki er ljóst hversu nýlegt stýrikerfið þarf að vera.
- Síminn þarf að vera nettengdur til að nota appið.
- Síminn þarf að vera með snertiskjá.

Performance Specs

- Ekki er ljóst hversu hraðvirk netssambandið þarf að vera né hversu hraðvirkur vélbúnaðurinn þarf að vera í tækinu.
- Æskilegt er að það taki appið ekki lengur en 5 sekúndur að staðfesta notanda upplýsingar.

Example: Backlog

Product backlog

| User Story | Sprint | Priority | Best case | Most likely case | Worst case | Expected case |
|---|--------|----------|-----------|------------------|------------|---------------|
| As a user I want to be able to create an event. | 1 | 10 | 3 | 5 | 10 | 6 |
| As a user I want to be able to view events | 1 | 10 | 5 | 8 | 13 | 8 |
| As a user I want to be able to search events by name and date | 1 | 10 | 1 | 2 | 5 | 2 |
| As a user I want to be able to make an account | 1 | 10 | 0.5 | 1 | 3 | 1 |
| As a user I want to be able to login to my account | 1 | 10 | 8 | 13 | 20 | 13 |
| As a user I want to be able to create a group, both private and public | 2 | 20 | 2 | 5 | 8 | 5 |
| As a user I want to be able to view groups | 2 | 20 | 1 | 3 | 5 | 3 |
| As a group member I want to be able to make a private event for other group members | 2 | 20 | 2 | 3 | 5 | 3 |
| As a group member I want to be able to add other users to the group | 2 | 20 | 1 | 2 | 4 | 2 |
| As a user I want to be able to search groups by name | 2 | 20 | 0.5 | 1 | 2 | 1 |
| As a user I want to be able to edit the information on my account | 3 | 30 | 0.5 | 0.5 | 1 | 1 |
| As a user I want to be able to mark myself as attending to an event | 3 | 30 | 2 | 3 | 5 | 3 |
| As a user I want to be able to see who is attending an event | 3 | 30 | 0.5 | 1 | 2 | 1 |
| As a user I want to be able to search user by username | 3 | 30 | 0.5 | 1 | 2 | 1 |
| As a user I want to be able to view the account of other users | 3 | 30 | 2 | 3 | 5 | 3 |
| As a user I want to be able to comment on events | 4 | 40 | 8 | 13 | 20 | 13 |



Example: Schedule

| School week | Date (from) | Date (to) | Project (our) | Project (code rev.) | Return assignment | Presentation | Sprint | milestone |
|-------------|-------------|------------|-------------------------|---------------------|-------------------|--------------|--------|--|
| 1 | 2019.01.07 | 2019.01.13 | requirements | | | | | |
| 2 | 2019.01.14 | 2019.01.20 | requirements | | | | | |
| 3 | 2019.01.21 | 2019.01.27 | requirements | | | | | |
| 4 | 2019.01.28 | 2019.02.03 | requirements | | ret. #1 | | | |
| 5 | 2019.02.04 | 2019.02.10 | design/core | | | expl. #1 | 1 | |
| 6 | 2019.02.11 | 2019.02.17 | design/core | | | | 1 | |
| 7 | 2019.02.18 | 2019.02.24 | design/core/programming | | ret. #2 | | | 1 design model, android studio core architecture, register, login, create chatroom |
| 8 | 2019.02.25 | 2019.03.03 | programming | | | expl. #2 | 2 | |
| 9 | 2019.03.04 | 2019.03.10 | programming | code rev. | | | 2 | |
| 10 | 2019.03.11 | 2019.03.17 | | | | | | |
| 11 | 2019.03.18 | 2019.03.24 | | | | | | |
| 12 | 2019.03.25 | 2019.04.01 | | | | | | |
| 13 | 2019.04.01 | 2019.04.07 | | | | | | |
| 14 | 2019.04.08 | 2019.04.14 | | | | | | |

The Gantt chart illustrates the project timeline across three months:

- February:** Sprint 1 (10-25), Sprint 2 (25-Mar 1).
- March:** Sprint 3 (1-25), Sprint 4 (25-Mar 1).
- April:** Sprint 5 (1-15).

Key milestones and activities marked on the timeline include:

- Design Model** (Week 5)
- Skeleton App** (Week 6)
- Running App & DB connection** (Week 7)
- Presentable App** (Week 8)
- Final App** (Week 9)
- Assignment 2** (Week 10)
- Code review** (Week 18)
- Assignment 3** (Week 25)
- Demo** (Week 15)
- Final Product** (Week 16)

Software Estimation

(continued)

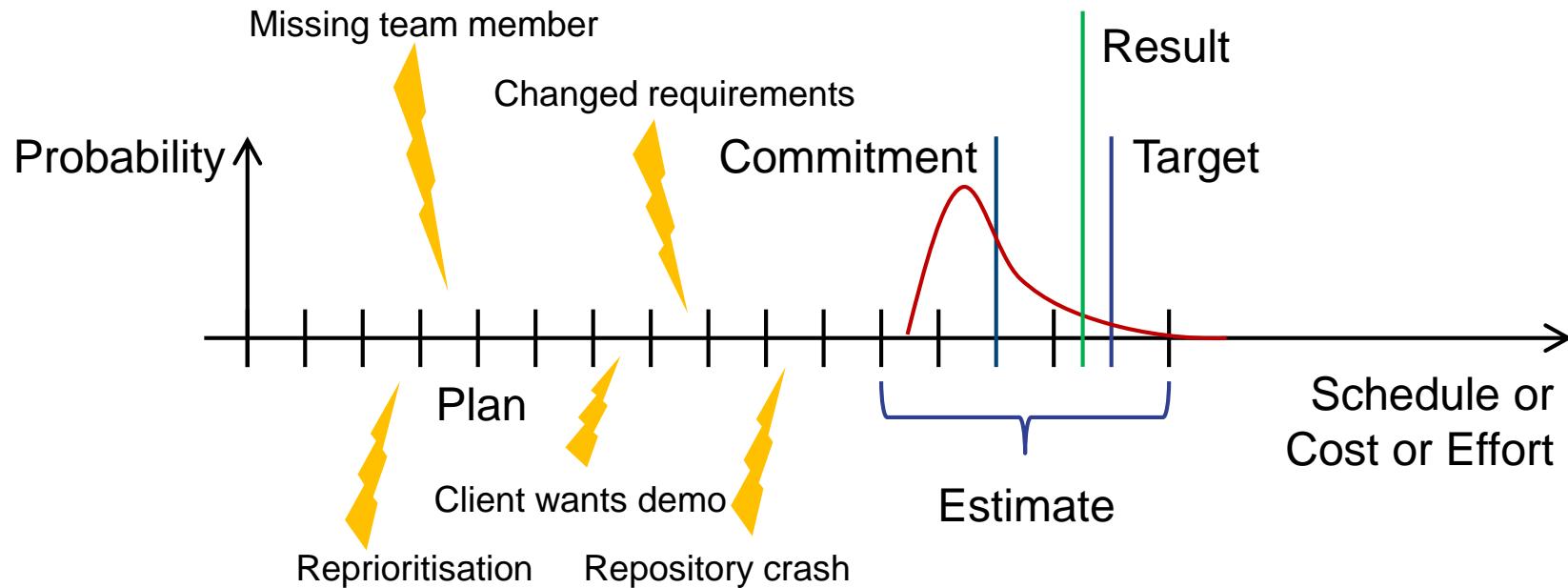
see also:

- McConnell: Software Estimation, Ch. 1, 3-5, 7-13
- Leffingwell: Agile Software Requirements, Ch. 8



Recap: Terminology

- **Target:** fixed value (deadline, budget etc.) determined by external factors
- **Estimate:** preliminary prediction of a future value/interval
- **Commitment:** Intention of reaching a certain future value
- **Plan:** Agreement of steps to achieve the commitment



In-Class Quiz #1 Solution: Projection from Counts



a) Counting defects late in project

- For the last 250 defects we fixed, we needed 2 hours per defect on average.
- 400 defects are currently open in our project.

➤ It should take us approximately $400 * 2 = 800$ hours to fix the open defects.

b) Counting web pages

- So far, each dynamic web page in the project cost us a total of 40 hours to design, code and test, on average.
- We have 10 web pages left to build.

➤ Under the assumption that we have built enough pages to have reliable experience, it should take us approximately $10 * 40 = 400$ hours to build the remaining pages.

Recap: Estimation Approaches Discussed in This Course

Motivation:

- The aim of effort estimates is **not to predict** the project result...
- ...but to judge whether the project target is **sufficiently realistic**..
- ...to be achievable through corresponding project management.

- A. Projection from counts
 1. Concrete items
 2. Function points
- B. Individual expert judgment
- C. Group expert judgment
 1. Wideband Delphi
 2. Planning Poker
- D. Qualitative estimation



Recap: Calculating Expected Cases

Weighted average, e.g.
ExpectedCase = (BestCase +
4 * MostLikelyCase + WorstCase) / 6

| Feature | Best Case | Most Likely Case | Worst Case | Expected Case |
|--------------|-----------|------------------|------------|---------------|
| 1 | 1.25 | 1.5 | 2.0 | 1.54 |
| 2 | 1.5 | 1.75 | 2.5 | 1.83 |
| 3 | 2.0 | 2.25 | 3.0 | 2.33 |
| 4 | 0.75 | 1 | 2.0 | 1.13 |
| 5 | 0.5 | 0.75 | 1.25 | 0.79 |
| 6 | 0.25 | 0.5 | 0.5 | 0.46 |
| 7 | 1.5 | 2 | 2.5 | 2.00 |
| 8 | 1.0 | 1.25 | 1.5 | 1.25 |
| 9 | 0.5 | 0.75 | 1.0 | 0.75 |
| 10 | 1.25 | 1.5 | 2.0 | 1.54 |
| Total | | | | 13.62 |

[Days to Complete]



Dangers of Adding Up Best / Worst Case Estimates

- Suppose we had the following estimates of weeks to complete 10 features:

| Feature | Best Case | Most Likely Case | Worst Case | Expected Case |
|---------|-----------------|------------------|-----------------|---------------|
| ... | ... | ... | ... | ... |
| Total | 20.0 | 28.3 | 38.6 | 28.62 |

- Adding up the individual estimates to obtain e.g. a “total best case” is pointless!
 - Suppose the probability of achieving any one of the best cases is 25% (i.e. $\frac{1}{4}$).
 - The probability of achieving any two of the best cases then is $\frac{1}{4} * \frac{1}{4} = 6.25\%$.
 - The probability of achieving the best case for all features is $\frac{1}{4}^{10} = 0.000095\%$.
 - This makes achieving the “total best case” completely unrealistic!**
- Caution:** Since single-point estimates are typically subconscious best-case estimates, adding those up to obtain a total estimate is particularly misleading!

B₁) Computing Probability Ranges for ≤10 Individual Estimates

1. Calculate standard deviation of the estimate totals:
 - StandardDeviation = (\sum WorstCaseEstimates – \sum BestCaseEstimates) / 6
2. Decide how confident your estimate needs to be
3. Pick according factor from table on the right, and calculate
 - PercentConfidentEstimate = ExpectedCase + Factor * StandardDeviation
 - Example:

| Feature | Best Case | Most Likely Case | Worst Case | Expected Case |
|--------------|-----------|------------------|------------|---------------|
| ... | ... | ... | ... | ... |
| Total | 20.0 | | 38.6 | 28.62 |

 [weeks]
 - Standard deviation = $(38.6 - 20.0) / 6 = 3.1$
 - 25% likely estimate: $28.62 - 0.67 * 3.1 \approx 27$ weeks
 - 75% likely estimate: $28.62 + 0.67 * 3.1 \approx 31$ weeks

These are *estimates* – don't kid yourself with unwarranted precision!

| Confid. | Factor |
|---------|--------|
| 2% | -2 |
| 10% | -1.28 |
| 16% | -1 |
| 20% | -0.84 |
| 25% | -0.67 |
| 30% | -0.52 |
| 40% | -0.25 |
| 50% | 0 |
| 60% | 0.25 |
| 70% | 0.52 |
| 75% | 0.67 |
| 80% | 0.84 |
| 84% | 1 |
| 90% | 1.28 |
| 98% | 2 |



Computing Probability Ranges

- Problem: Statistics of previous approach assume that 99.7% of actual results fall into estimated range (between best and worse case).
 - i.e. only 3 out of 1000 estimates could fall outside estimated range!
- Obviously completely unrealistic
 - Most people are actually about 30% sure when they think they are 90% sure
 - With practice, about 70% can be achieved
- For ≤ 10 estimated items, the impact of this error is negligible, so we can still use the previous simple approach.
- For > 10 estimated items, the impact becomes more pronounced, so we need to use a somewhat more complex approach to calculate the standard deviation:

B₂) Computing Probability Ranges for >10 Individual Estimates

1. Check how many of your actual outcomes fall within your estimation range
2. Determine according divisor in table on the right
3. Calculate the variance of each *individual* estimate:
 - Variance = ((WorstCaseEstimate – BestCaseEstimate) / Divisor)²
4. Calculate total of the individual variances
5. Take the total's square root to obtain the standard deviation
6. Continue with step 2 of previous approach

| % in range | Divisor |
|------------|---------|
| 10% | 0.25 |
| 20% | 0.51 |
| 30% | 0.77 |
| 40% | 1 |
| 50% | 1.4 |
| 60% | 1.7 |
| 68% | 2 |
| 70% | 2.1 |
| 80% | 2.6 |
| 90% | 3.3 |
| 99.7% | 6 |

Example: Percentage Confident Estimate

- Based on data we saw earlier, this team's results fall into the estimated range 80% of the time
 - Use divisor 2.6 to calculate variances of individual estimates
- Standard deviation:
 - $\sqrt{6.48} = 2.55$
- 25% likely estimate:
 - $28.62 - 0.67 * 2.55 \approx 27$ weeks
- 75% likely estimate:
 - $28.62 + 0.67 * 2.55 \approx 30$ weeks

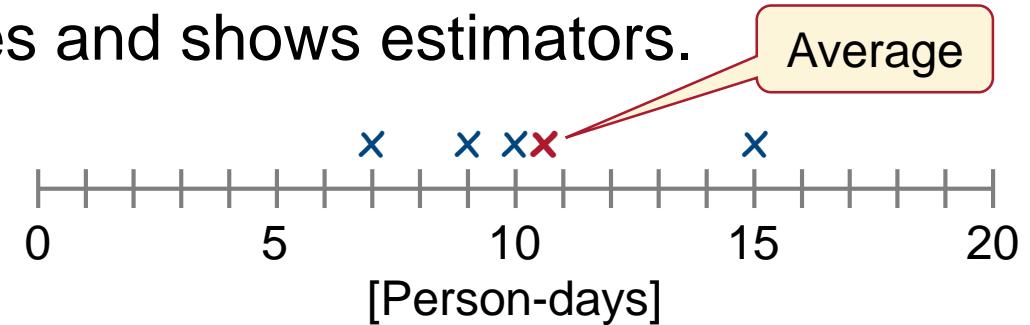
| Feature | Best Case | Most Likely Case | Worst Case | Expected Case | Variance |
|---------|-----------|------------------|------------|---------------|----------|
| 1 | 1.6 | 2.0 | 3.0 | 2.10 | 0.290 |
| 2 | 1.8 | 2.5 | 4.0 | 2.63 | 0.716 |
| 3 | 2.0 | 3.0 | 4.2 | 3.03 | 0.716 |
| 4 | 0.8 | 1.2 | 1.6 | 1.20 | 0.095 |
| 5 | 3.8 | 4.5 | 5.2 | 4.50 | 0.290 |
| 6 | 2.2 | 5.0 | 3.4 | 4.97 | 0.716 |
| 7 | 2.2 | 2.4 | 3.4 | 2.53 | 0.213 |
| 8 | 0.8 | 1.2 | 2.2 | 1.30 | 0.290 |
| 9 | 1.6 | 2.5 | 3.0 | 2.43 | 0.290 |
| 10 | 1.6 | 4.0 | 6.0 | 3.93 | 2.864 |
| Total | | | | 28.62 | 6.48 |

[weeks to complete]



C₁) Group Expert Judgment: Wideband Delphi

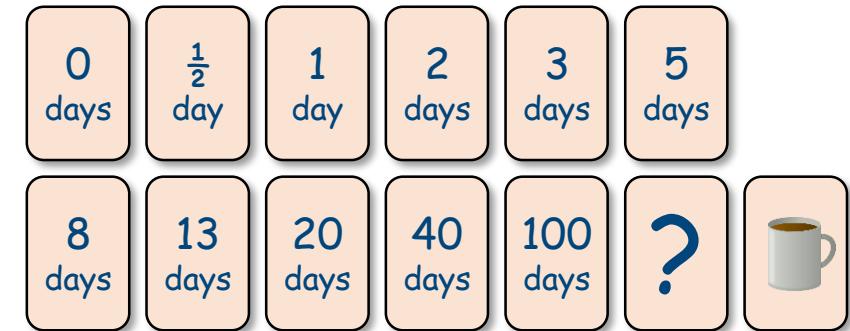
1. Delphi coordinator presents each estimator with specification, estimation form
2. Estimators prepare initial estimates for features individually.
3. Coordinator calls group meeting in which estimators discuss estimation issues.
4. Estimators give their individual estimates to coordinator anonymously.
5. Coordinator prepares summary of estimates and shows estimators.



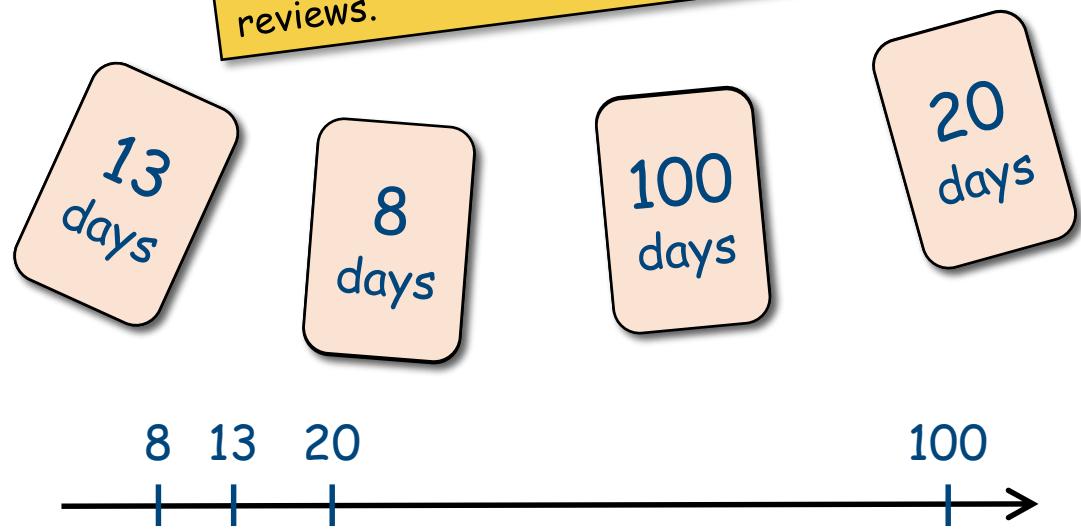
6. Coordinator lets estimators discuss variations in estimates.
 7. Estimators vote anonymously on whether to accept average or return to #3.
- Final estimate is a *range* created in discussion, average is expected case.

C₂) Group Expert Judgment: Planning Poker (Recap)

1. Place a user story in the middle of the table.
2. Everyone picks an estimate for the story from their hand and places the card face-down on the table.
3. Everyone turns their card over at the same time.
4. Note and discuss the spread of the estimates.
5. Decide on an estimate that the whole team can agree on.
 - Not necessarily the average!



As a website visitor, I want to rank reviews I read, in order to help other visitors to find the most helpful reviews.



What to Estimate in Planning Poker?

Story Points

- Synthetic measure
- Only indicates relative complexity (and therefore relative effort) of user stories, but does not directly translate into absolute effort
 - Translation into absolute effort is implicit as team executes sprint and achieves a certain velocity
- Estimate focuses on complexity and is kept free of real-time considerations
 - more objective, but hard to communicate

Person-Days

- Describes an actual property
 - Easier to handle by teams who are new to agile approach
 - easier to communicate to management
 - Estimates may be politically biased
 - Developers may not dare to estimate high so as not to appear “slow”
-
- **Suggestion:** Use story points, but have informal convention that 1 story point corresponds to 1 person-day

D) Qualitative Estimation: T-Shirt Sizing

- Chicken and egg problem in agile projects:
 - Team members shouldn't spend effort on understanding / refining a feature's requirements until they are reasonably sure it is actually going to be built
 - Management doesn't want to decide on whether a feature shall be built until they have some idea of the required effort
- Solution:
 - Initially use a highly intuitive, relative, coarse and fast estimation scheme
 - Classify features by Business Value and Development Cost
 - using "t-shirt sizes" (small, medium, large, extra large) as categories
- Example:

| Feature | Business Value | Development Cost |
|---------|----------------|------------------|
| A | Large | Small |
| B | Small | Large |
| C | Large | Large |
| ... | ... | ... |

Calculating Net Business Value Based on T-Shirt Sizes

- Discussion on which features to retain and which to cut is easier when feature list is sorted according to cost/benefit evaluation
- Use following matrix (or similar one of your own) to assign **net business value**:

| Net Business Value | | Development Cost | | | |
|--------------------|-------------|------------------|-------|--------|-------|
| Business Value | Extra Large | Extra Large | Large | Medium | Small |
| | Large | 0 | 4 | 6 | 7 |
| | Medium | -4 | 0 | 2 | 3 |
| | Small | -6 | -2 | 0 | 1 |
| | Small | -7 | -3 | -1 | 0 |

- Sort features by resulting net business value
- Identify “definite yes” and “definite no” features, discuss others in more detail

Example: Calculating Net Business Value

| Feature | Business Value | Development Cost | Net Business Value |
|---------|----------------|------------------|--------------------|
| A | L | S | 3 |
| F | L | M | 2 |
| C | L | L | 0 |
| D | M | M | 0 |
| G | S | S | 0 |
| I | S | S | 0 |
| H | S | M | -1 |
| E | M | L | -2 |
| ... | ... | ... | ... |
| B | S | L | -3 |

Definite YES
area

Up for
discussion

Definite NO
area

When to Use Which Approach?

- **Use rough qualitative estimation, e.g. T-Shirt Sizing (D)**

- WHEN: Initial stages of a project
- WHY: To make initial decisions on the general scope of project
- BASIS: General vision of project, high-level business and technology constraints
- ACCURACY: Quite reliable in definitive YES and NO areas, very fuzzy in the middle region
- REFINEMENT: Refine included features in product and sprint backlog discussions (B, C)

- **Use projection from counts, e.g. average effort per [item] (A)**

- WHEN: Early stages of a project
- WHY: To get an initial rough idea of the overall size, effort and schedule of the project
- BASIS: Reasonably complete overview of requirements, comparable historic data
- ACCURACY: Highly dependent on accuracy of data, will be deprecated by later fluctuations
- REFINEMENT: Use more precise approaches for product / sprint backlog estimates (B, C)

When to Use Which Approach?

- **Use individual expert judgment, e.g. Three-Point Estimates (B)**
 - WHEN: Early stages of a project
 - WHY: To obtain initial estimates for product backlog items, based on different scenarios
 - BASIS: Discussion of requirements, experts' experience
 - ACCURACY: Higher than “gut feelings” that otherwise go into group expert estimate (C_1)
 - REFINEMENT: Individual sprint planning (C_2) will make more informed short-term estimates
- **Use elaborate group expert judgment, e.g. Wideband Delphi (C_1)**
 - WHEN: Early stages of a project
 - WHY: To obtain initial estimates for product backlog items, based on prior experience
 - BASIS: Discussion of requirements, experts' experience
 - ACCURACY: Reasonably high if experts have sufficient domain knowledge
 - REFINEMENT: Individual sprint planning (C_2) will make more informed short-term estimates

When to Use Which Approach?

- **Use pragmatic group expert judgment, e.g. Planning Poker (C_2)**
 - WHEN: Sprint planning meetings
 - WHY: To obtain detailed estimates for all sprint backlog items and commit to sprint contents
 - BASIS: Elaboration of and discussion of requirements, breakdown of tasks
 - ACCURACY: High w enough domain knowledge, performance experience, breakdown detail
 - REFINEMENT: Tracking velocity helps team to calibrate estimates in subsequent sprints



In-Class Quiz #2: Estimation Methods

- Indicate which approach you would use in the following situations:
 - a) Establishing a rough project scope
 - b) Establishing first release date (new project, new domain)
 - c) Establishing next release date (project you've been on for a year)
 - d) Getting an idea of whether you can still complete a user story before end of sprint
 - e) Sprint planning

ID: _____ @hi.is Date: _____
a) _____ d) _____
b) _____ e) _____
c) _____



- Possible choices:

- (A) Projection from counts
- (B) Three-Point Estimates
- (C₁) Wideband Delphi
- (C₂) Planning Poker
- (D) T-Shirt Sizing

Advantages of Precise Estimates

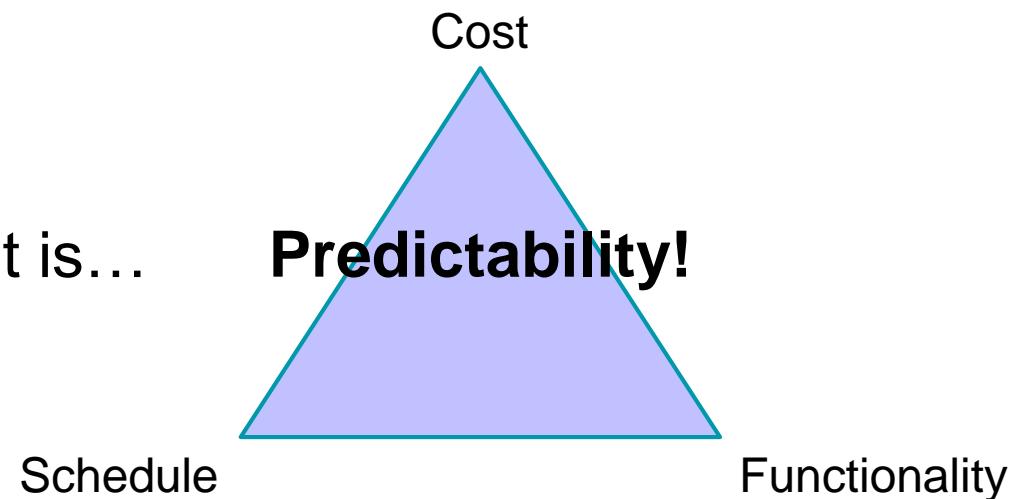
- Confirmation that project is on track
- Higher software quality due to reduced stress
- Better alignment with plans of other business units
- Higher financial safety
- Reference values for later projects
- More competent image of developer team
- Early risk indicators when target and estimate diverge
 - i.e. early chance to remedy the situation...
 - Changed project scope
 - More staff
 - Use of experts
 - Incremental deliverables
 - Decision against project
 - ...instead of ignoring/negotiating the estimate and accepting the risk of
 - Schedule or budget overruns
 - Forced functionality reductions



Project Qualities Expected by Customers

- Every customer prefers a project that is
 - on schedule
 - within budget
 - fulfilling expectations
- If you need to compromise on any of these,
 - let the customer know early what to expect
 - and work on an acceptable solution together

➤ What customers value most is...



Practicing Estimation and Controlling Techniques

- Try to re-estimate your project based on a more refined list of user stories, using one of the methods presented in class
 - Check how the results of different estimation methods compare
 - Gain experience with the estimation process
 - Throughout the project, record your working hours so you can compare them to your estimates later
 - Gain experience with how long you need for certain tasks
 - Get a feeling for how your actual performance differs from your estimated performance
 - Throughout the project, maintain a burndown chart recording your progress through the sprints
 - Gain experience with velocity tracking
 - Get a feeling for how much you can commit to, and what overhead you need to consider
- **In your final presentation (Ass. 4), present and discuss estimated vs. actual efforts and burndown chart as part of your project retrospective**
- Your quantitative performance will not be graded, but your critical reflection will be





Hugbúnaðarverkefni 2 / Software Project 2

5. Android Development Basics

HBV601G – Spring 2020

Matthias Book



HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRVÍSINDASVIÐ
ÍÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD

In-Class Quiz Prep

- Please prepare a small scrap of paper with the following information:

ID: _____ @hi.is Date: _____

a) _____

e) _____

b) _____

f) _____

c) _____

g) _____

d) _____

- During class, I'll show you questions that you can answer very briefly
 - Just numbers or letters, no elaboration
- Hand in your scrap at the end of class
- All questions in a quiz weigh same
- All quizzes (ca. 10-12 throughout semester) have the same weight
 - Your worst 2 quizzes will be disregarded
- Overall quiz grade counts as optional question worth 7.5% on final exam



In-Class Quiz 2 Solution



- Indicate which approach you would use in the following situations:
 - a) Establishing a rough project scope
 - b) Establishing first release date (new project, new domain)
 - c) Establishing next release date (project you've been on for a year)
 - d) Getting an idea of whether you can still complete a user story before end of sprint
 - e) Sprint planning
- Best answers:
 - (D) T-Shirt Sizing
 - (B) Three-Point Estimates
 - (C₁) Wideband Delphi
 - (A) Projection from counts
 - (C₂) Planning Poker

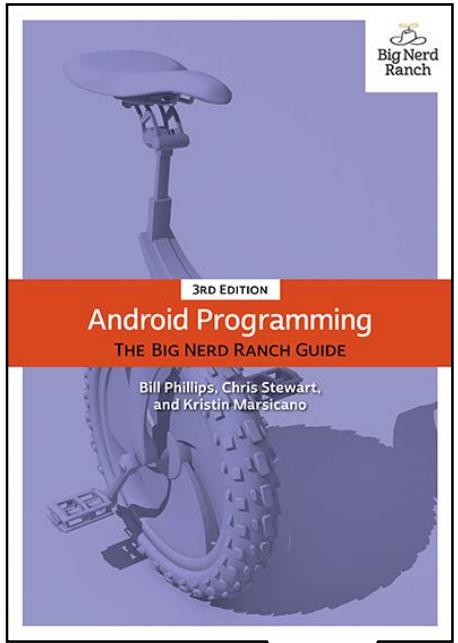
Update: Assignment 2 – Design Model

Note: Clarified from version shown in first week.

- By the deadline specified in your project plan, submit a **design model** in Uglá:
 - UML class diagram of your system (detail level reflecting state of implementation/planning)
 - Clearly distinguish server- and client-side components
 - For client (Android) side, show only Model and Controller classes, not View classes
 - Suitable UML behavioral diagrams to show:
 - User navigation in your app
 - Control flow between key components (**within app, and between client and server**)
- On the Monday after submission, present and **explain** your model to your tutor:
 - How will your system work? What influenced your design choices? What's still unknown?
- **Grading criteria** (25% of this assignment's grade each):
 - System structure is plausible, consistent with requirements and behavioral diagrams
 - User navigation is plausible and shown in a suitable diagram
 - Control flow is plausible and shown in a suitable diagram
 - UML diagrams are clean and syntactically correct



A Simple Android App



see also:

- Phillips et al.: Android Development, Ch. 1



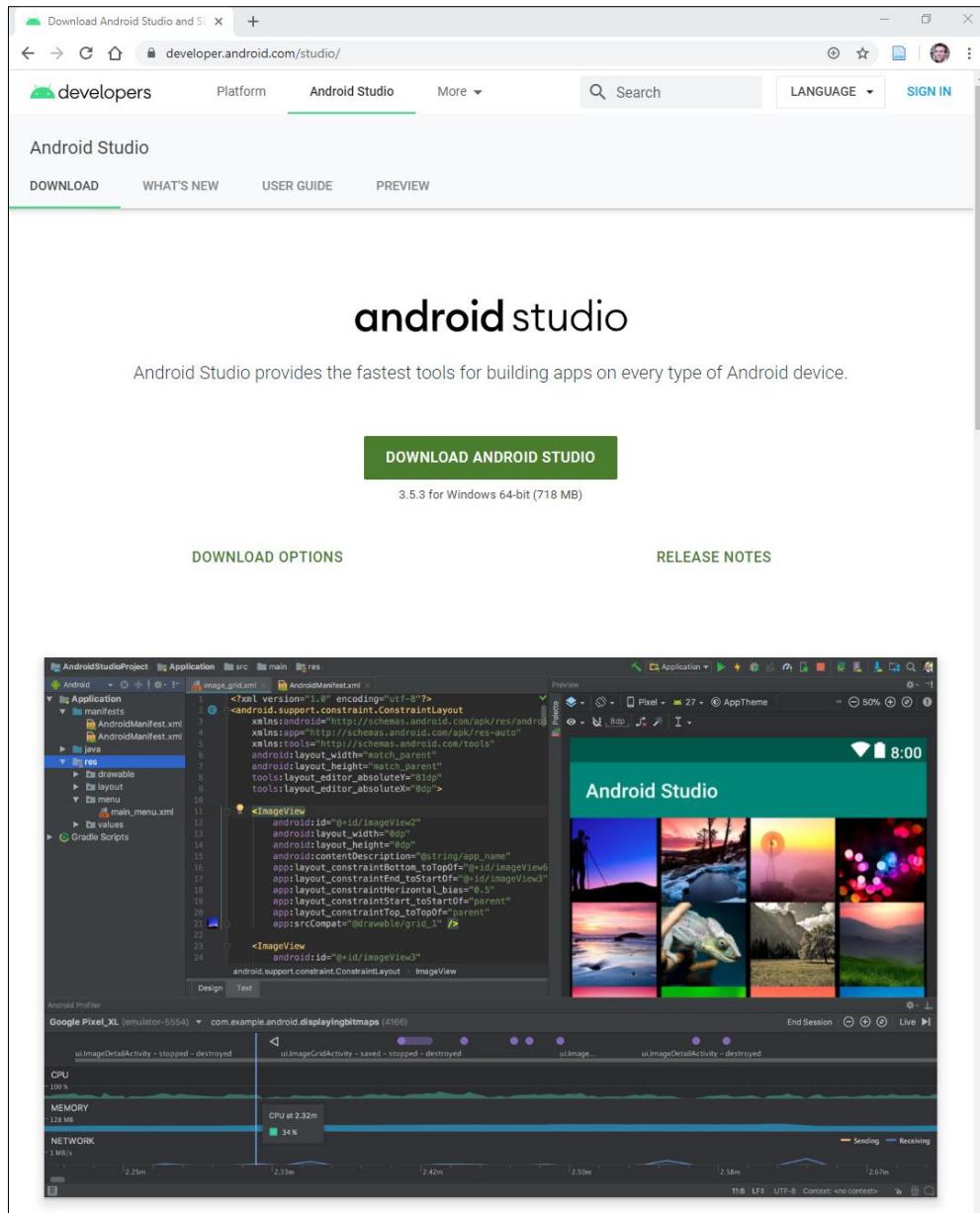
Development Environment

■ Android Studio

- IDE based on IntelliJ
- Android SDK (class library)
- Tools for debugging and testing apps
- Android device emulators
- Download: <https://developer.android.com/studio/>
 - Caution: 718+588+733 MB download, >2-3 GB install

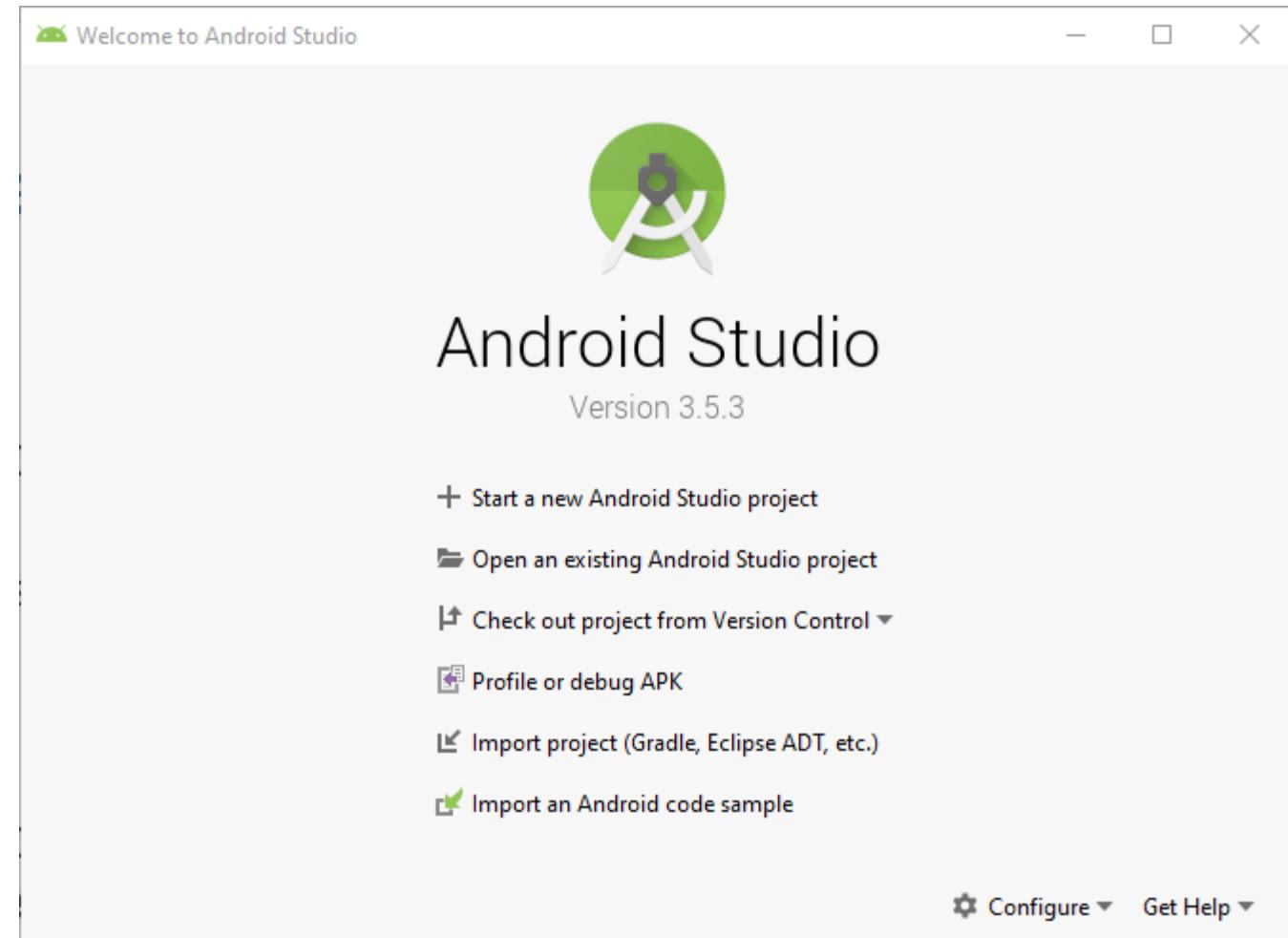
■ A physical Android device for testing

- Alternatively, use the included emulators



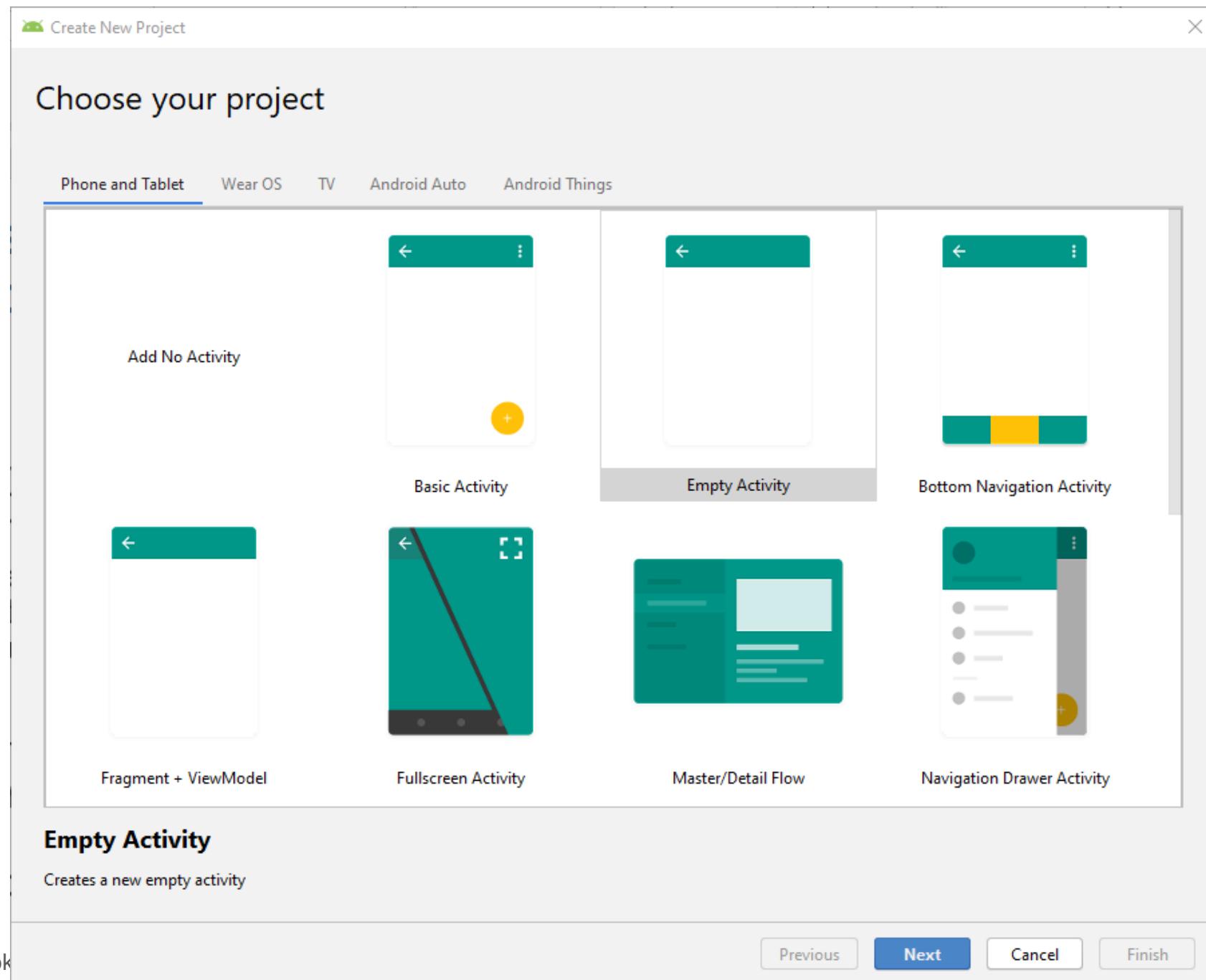
Creating a Project

- Start Android Studio
- Choose “Start a new Android Studio project”
 - If the welcome screen doesn’t appear, use “File > New > New Project...” menu



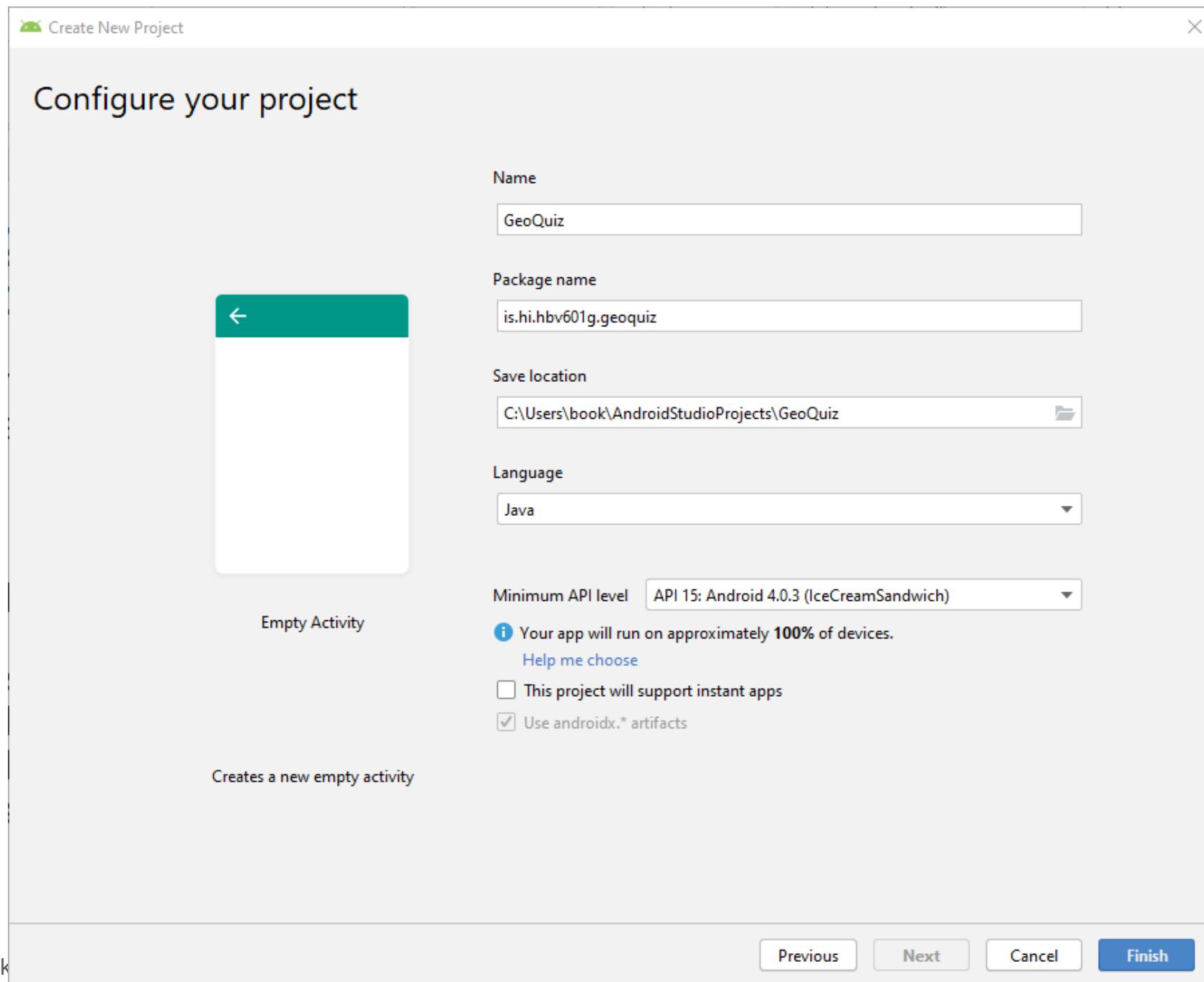
Choose a Project Type

- Use “Empty Activity” for starters
 - Other activities contain extra features
- Note that activities for many different device types are available



Configure Project

- You'll have to choose an API level
- Your app will not be available to devices running an Android version with a lower API level, so choose wisely



Choose API Level

- Pick a reasonable balance between required device functionality and desired market share

Android Platform/API Version Distribution

| ANDROID PLATFORM VERSION | API LEVEL | CUMULATIVE DISTRIBUTION |
|--------------------------|-----------|-------------------------|
| 4.0 Ice Cream Sandwich | 15 | |
| 4.1 Jelly Bean | 16 | 99.6% |
| 4.2 Jelly Bean | 17 | 98.1% |
| 4.3 Jelly Bean | 18 | 95.9% |
| 4.4 KitKat | 19 | 95.3% |
| 5.0 Lollipop | 21 | 85.0% |
| 5.1 Lollipop | 22 | 80.2% |
| 6.0 Marshmallow | 23 | 62.6% |
| 7.0 Nougat | 24 | 37.1% |
| 7.1 Nougat | 25 | 14.2% |
| 8.0 Oreo | 26 | 6.0% |
| 8.1 Oreo | 27 | 1.1% |

Ice Cream Sandwich

- Contacts Provider**
- Social APIs
- User profile
- Invite intent
- Large photos

Accessibility

- Explore-by-touch mode
- Accessibility for views
- Accessibility services
- Improved text-to-speech engine support

User Interface

- Spell checker services
- Improved action bar
- Grid layout
- Texture view
- Switch widget
- Improved popup menus
- System themes
- Controls for system UI visibility
- Hover event support
- Hardware acceleration for all windows

Voicemail Provider

- Add voicemails to the device

Multimedia

- Media effects for images and videos
- Remote control client
- Improved media player

Camera

- Face detection
- Focus and metering areas
- Continuous auto focus
- Camera broadcast intents

Enterprise

- VPN services
- Device policies
- Certificate management

Device Sensors

- Improved sensors
- Temperature sensor
- Humidity sensor

<https://developer.android.com/about/versions/android-4.0.html>

OK Cancel

GeoQuiz > app > src > main > res > layout > activity_main.xml app No devices

Android Project Resource Manager I: Project I: Favorites I: Structure I: Favorites I: Favorites

1: Project 2: Favorites

activity_main.xml MainActivity.java

<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:and
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="match_parent"
tools:context=".MainActivity">

<TextView
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:text="Hello World!"
 app:layout_constraintBottom_toBottomOf="parent"
 app:layout_constraintLeft_toLeftOf="parent"
 app:layout_constraintRight_toRightOf="parent"
 app:layout_constraintTop_toTopOf="parent"/>

Preview Assistant Gradle Preview

Click on code to highlight corresponding widget (and vice versa)

Browse project files

Switch between code and design view

Toggle tool windows

Device File Explorer

Build: Sync

GeoQuiz: synced successfully at 05-Feb-20 11:05

Download https://services.gradle.org/distributions/gradle-5.4.1-all.zip

Starting Gradle Daemon

Run build C:\Users\book\Andro

Load build

Configure build

Calculate task graph

Run tasks

1 m 21 s 237 ms
11 s 565 ms
1 s 125 ms
57 s 192 ms
2 s 757 ms
52 s 453 ms
73 ms
1 s 268 ms

Event Log

TODO Terminal Build 6: Logcat

Matthias Book: Software Project 2

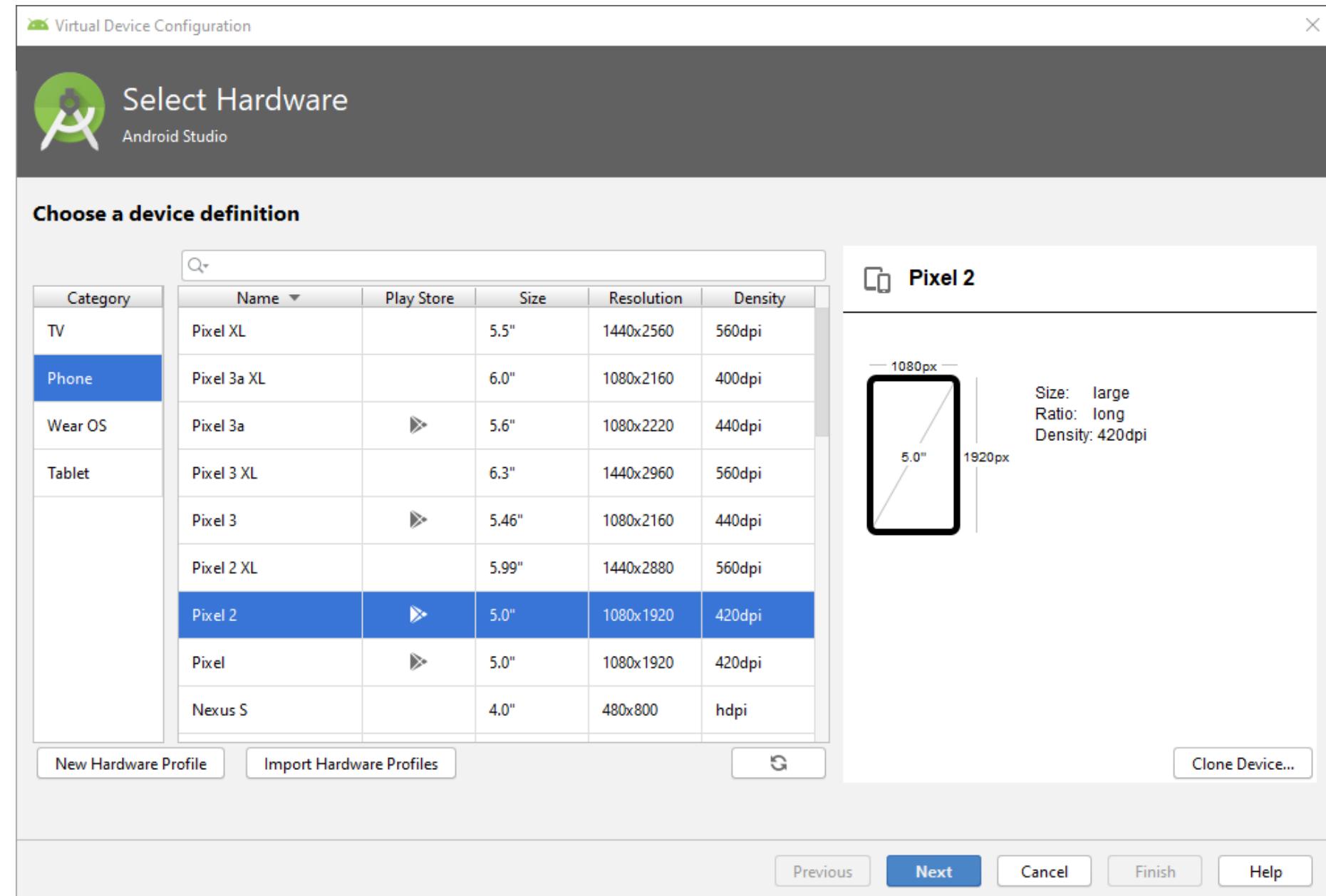
* daemon started successfully (5 minutes ago)

1:1 CRLF UTF-8 4 spaces

Android Studio Overview

Prepare an Emulator

- Open menu “Tools > AVD Manager”
- Click “Create Virtual Device...” button
- Pick a device of your choice



Choose a System Image

- Pick an operating system image of your choice
 - API level at least as high as your project's API level
- Need to download image before proceeding

Virtual Device Configuration

System Image

Select a system image

Recommended x86 Images Other Images

| Release Name | API Level | ABI | Target |
|-----------------|-----------|-----|-----------------------------|
| Q Download | 29 | x86 | Android 10.0 (Google Play) |
| Pie Download | 28 | x86 | Android 9.0 (Google Play) |
| Oreo Download | 27 | x86 | Android 8.1 (Google Play) |
| Oreo Download | 26 | x86 | Android 8.0 (Google Play) |
| Nougat Download | 25 | x86 | Android 7.1.1 (Google Play) |
| Nougat Download | 24 | x86 | Android 7.0 (Google Play) |

Oreo

API Level
26
Android
8.0
Google Inc.

System Image
x86

Recommendation
VT-x is disabled in BIOS.
Troubleshoot

We recommend these Google Play images because this device is compatible with Google Play.

Questions on API level?

A system image must be selected to continue.

Previous Next Cancel Finish Help



Configure the Virtual Device

The image shows two overlapping windows from the Android Studio interface. The top window is titled "Virtual Device Configuration" and "Android Virtual Device (AVD) - Android Studio". It displays the "Verify Configuration" screen with various settings for the AVD named "Pixel 2 API 26". The bottom window is titled "AVD Name" and shows the configuration details for the same AVD. The "AVD Name" window includes sections for "Memory and Storage", "Device Frame", "Keyboard", and "Hide Advanced Settings". A note at the bottom of the "AVD Name" window says "How do I create a custom hardware skin?". At the bottom right of both windows are buttons for "Previous", "Next", "Cancel", "Finish", and "Help".

Virtual Device Configuration

Android Virtual Device (AVD)

Android Studio

Verify Configuration

AVD Name Pixel 2 API 26

AVD Id Pixel_2_API_26

Pixel 2 5.0 1080x1920 xxhdpi Change...

Oreo Android 8.0 x86 Change...

Startup orientation Portrait Landscape

Camera Front: Emulated Back: VirtualScene

Network Speed: Full Latency: None

Emulated Performance Graphics: Automatic Boot option: Cold boot Quick boot Choose from snapshot (no snapshots) Multi-Core CPU 2

AVD Name

The name of this AVD.

Memory and Storage RAM: 1536 MB VM heap: 256 MB Internal Storage: 2048 MB SD card: Studio-managed 512 MB External file

Device Frame Enable Device Frame Custom skin definition pixel_2 How do I create a custom hardware skin?

Keyboard Enable keyboard input

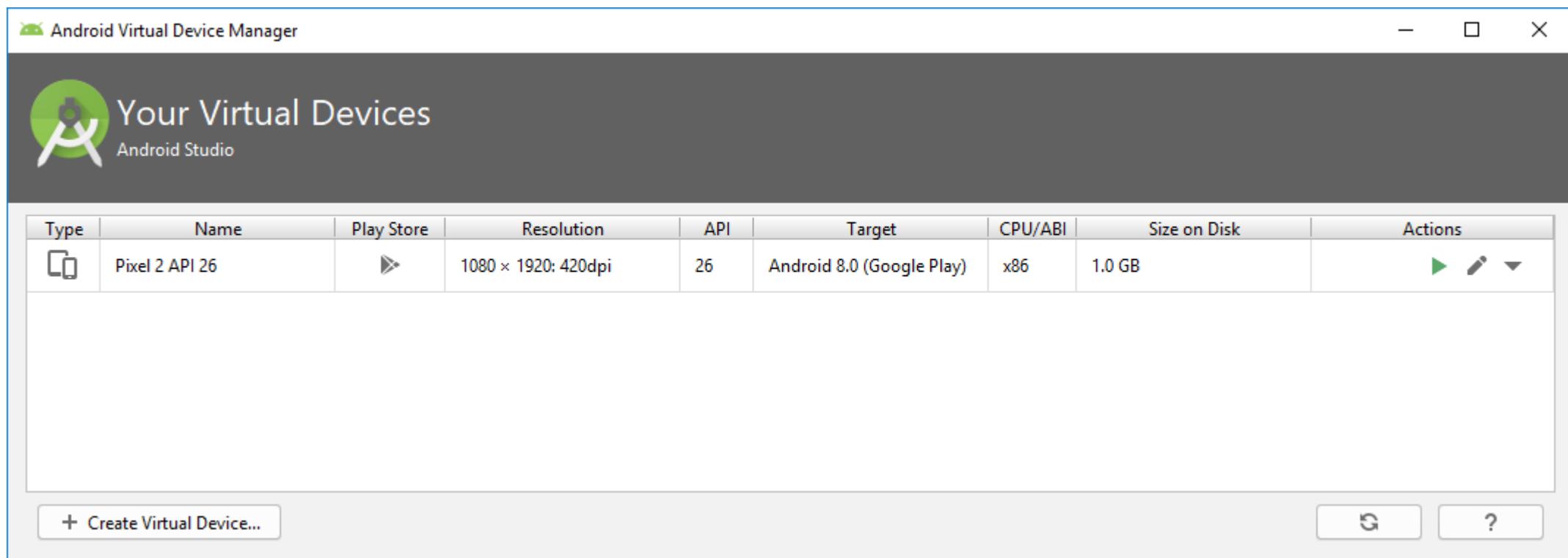
Hide Advanced Settings

Previous Next Cancel Finish Help

- For now, just name it

Managing Virtual Devices

- Your new device shows up in the AVD Manager
 - Use the dropdown menu on the right to manage your virtual devices
- Start virtual device by clicking the ▶ icon



Working With the Virtual Device

- Boot-up takes about a minute
 - Keep running for later tests

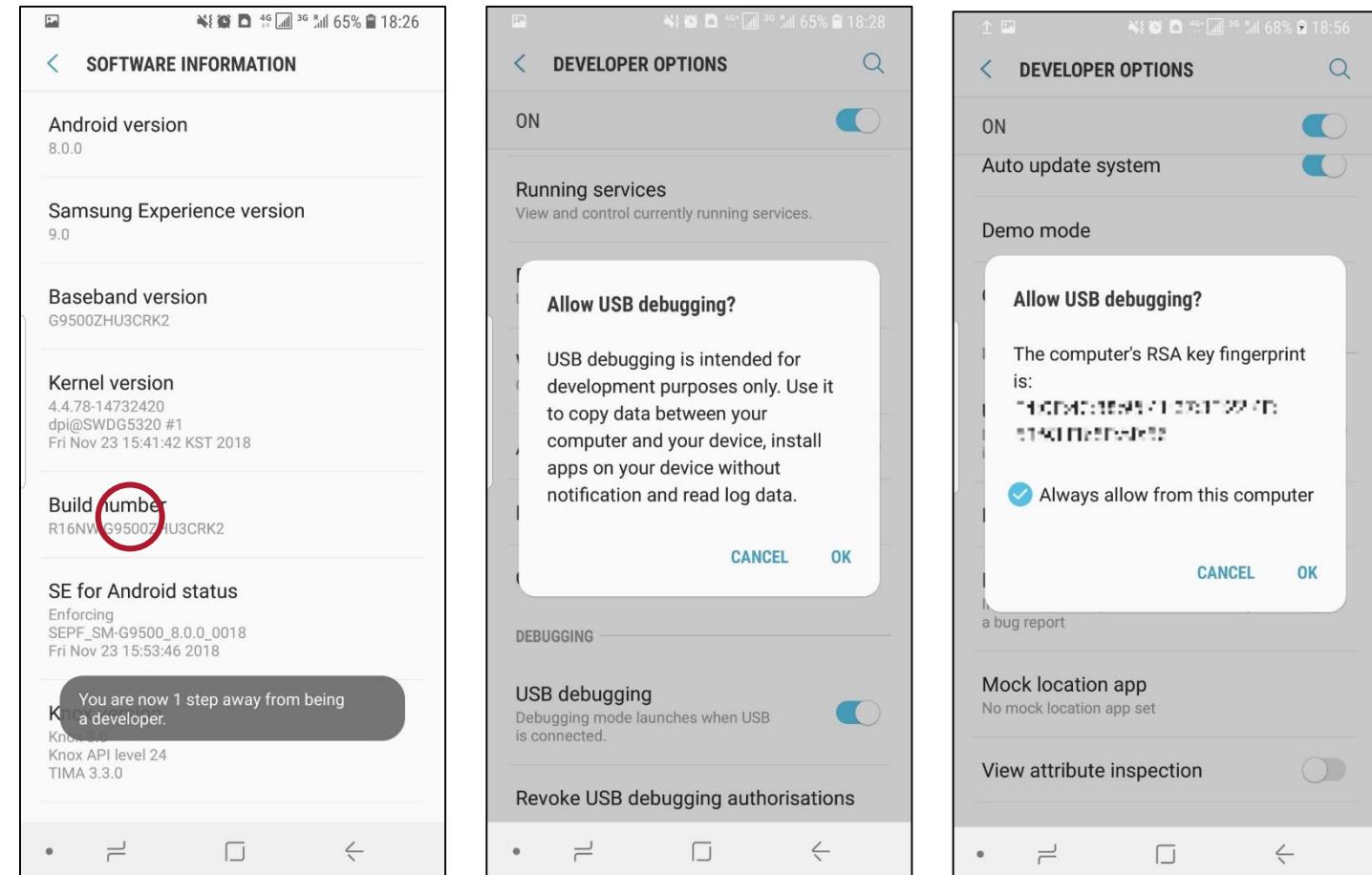


- Emulate sensor inputs etc. by opening “...” menu

A screenshot of the 'Extended controls' window for a Nexus_5X API 26 virtual device. The window title is 'Extended controls - Nexus_5X_API_26:5554'. On the left is a sidebar with icons for Location, Cellular, Battery, Camera, Phone, Directional pad, Microphone, Fingerprint, Virtual sensors, Bug report, Snapshots, Screen record, Google Play, Settings, and Help. The main area is titled 'GPS data point' and shows a coordinate system set to 'Decimal'. The latitude is 37.422 and the longitude is -122.084. Below this is a section for 'Currently reported location' with the same coordinates and altitude (5.0), speed (0.0), and heading (0.0). There is a 'GPS data playback' section with columns for Delay (sec), Latitude, Longitude, Elevation, Name, and Description. At the bottom are buttons for 'SEND', 'LOAD GPX/KML', and a speed control set to 'Speed 1X'.

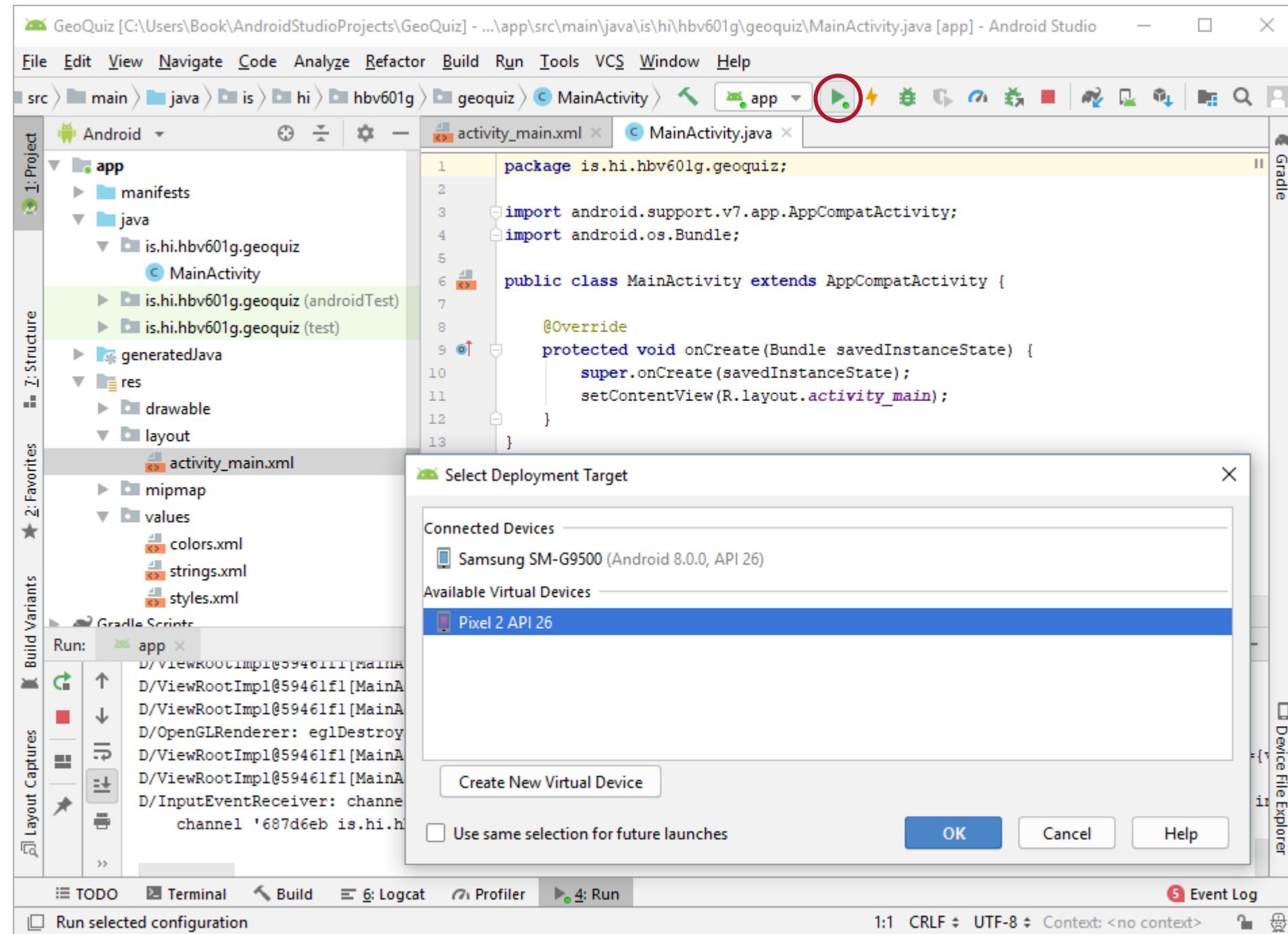
Preparing a Physical Device

- To deploy apps on a physical device, enable USB debugging on that device:
 - Go to “Settings > About Phone > Software Information” (or similar)
 - Press “Build Number” seven times
 - Go to “Settings > Developer options” (was invisible before)
 - Enable “USB debugging”
 - Confirm that USB debugging shall be allowed in general
- Then connect device to computer using USB cable
 - Authorize computer for USB debugging



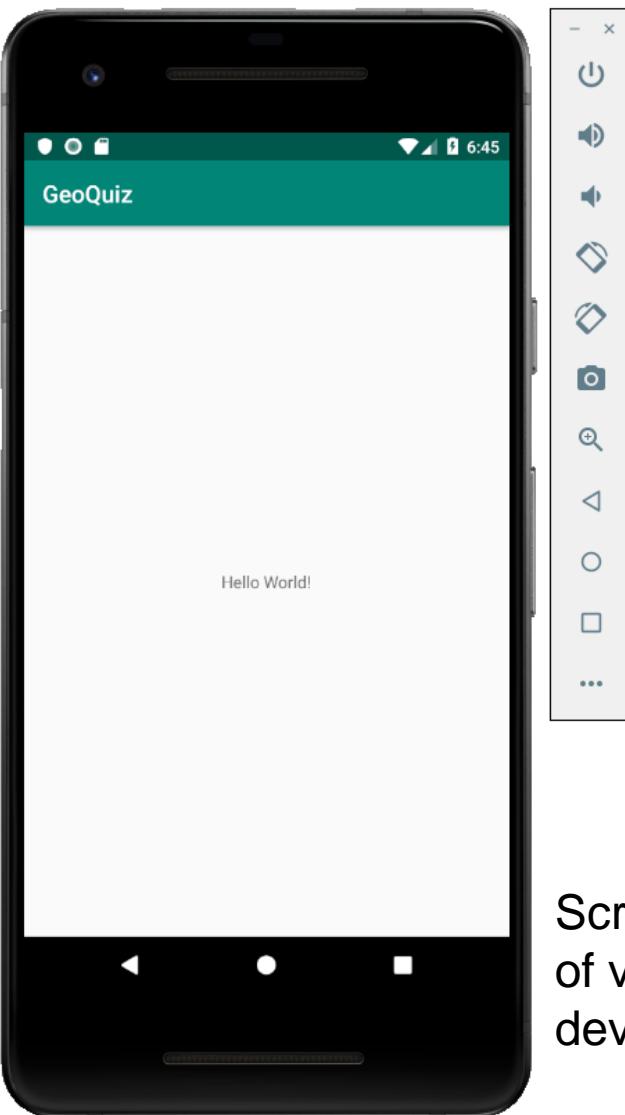
Running the App

- Click on green ► icon
 - or press Ctrl+R
- Pick desired device (virtual or physical, if connected)
- App will be deployed and started on device

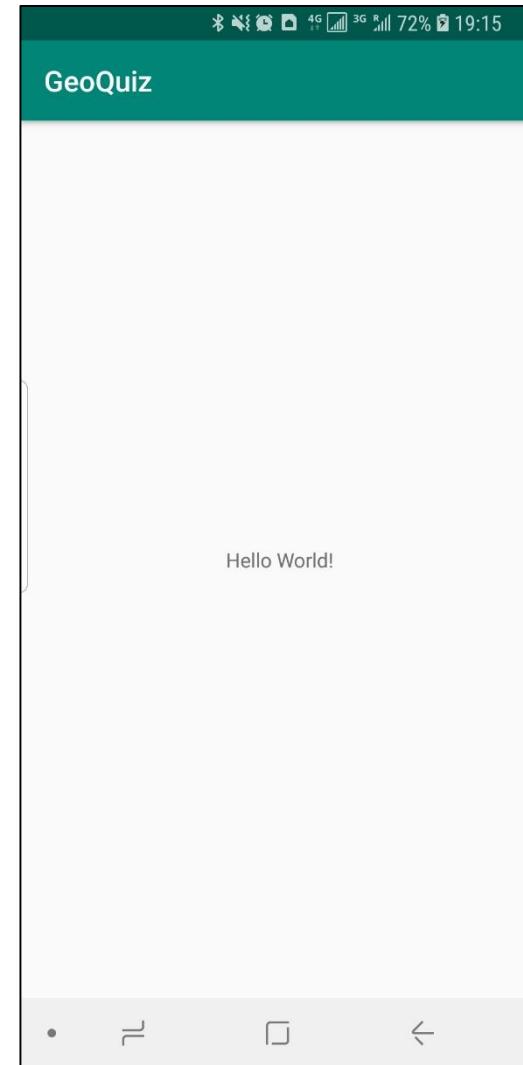


Testing the App

- App will start on device
- Interact as desired
 - Use virtual device sidebar for emulation of physical sensor input (e.g. rotation, camera, location...)
- Leave app using Back button to stop execution



Screenshot
of virtual
device



Screenshot
from physical
device

Basic Project Structure

- **Java classes** go into `\app\src\main\java`
 - Most importantly: Activity classes
- **Resource files** go into `\app\src\main\res`
 - Most importantly: Layout files in ...`\layout`
- Naming convention:
 - Activities end with **Activity**
 - Corresponding layouts start with **activity_**

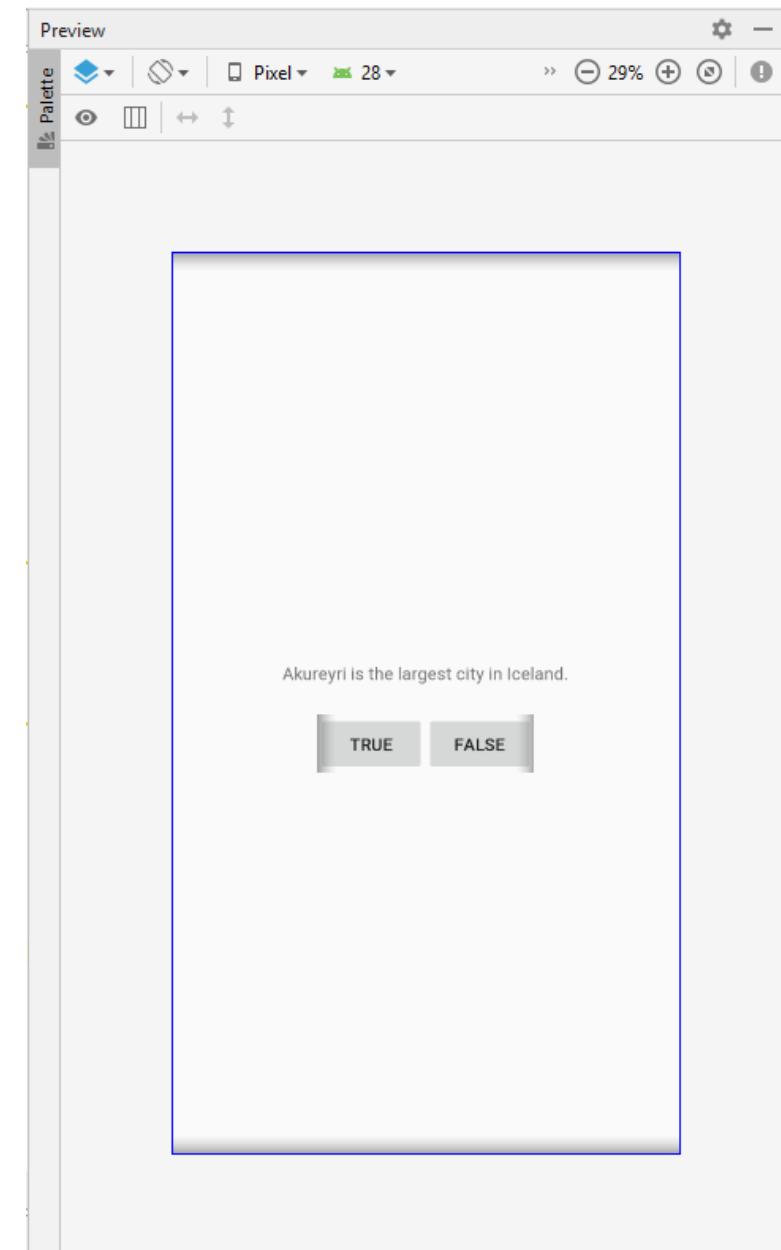
The screenshot shows the Android Studio interface with the following details:

- Project Tree:** Shows the project structure under `GeoQuiz > app > src > main > java > is > hi > hbv601g > geoquiz > MainActivity`.
- Main Activity Java File:** `MainActivity.java` is selected in the code editor.
- Code Content:**

```
1 package is.hi.hbv601g.geoquiz;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5
6 public class MainActivity extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_main);
12     }
13 }
14
15 }
16 }
```

Activities and Layouts

- An **activity** is responsible for managing user interaction with a screen of information, i.e. to implement a part of the functionality of your app.
 - Activities are implemented in subclasses of **Activity**
 - A **layout** defines a set of user interface (UI) objects and their screen positions.
 - Layouts are defined in XML files stored in **app/src/main/res/layout**
 - Each XML element in the file corresponds to one UI widget.
- The activity **MainActivity** manages the UI that the layout **activity_main.xml** defines.
- Example:
 - **activity_main.xml** defines where the “TRUE” and “FALSE” buttons are placed and how they are labeled.
 - **MainActivity** implements what happens when one of the buttons is clicked.



Defining a Layout

```
activity_main.xml
MainActivity.java
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

    <TextView
        android:text="@string/question_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp"/>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:id="@+id/true_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/true_button"/>

        <Button
            android:id="@+id/false_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/false_button"/>
    </LinearLayout>
</LinearLayout>
```

Preview

Palette

Pixel 2 28 22%

Design Text

MASKOTT ISLANDS

Matthias Book: Software Project 2

LinearLayout
(vertical orientation)

TextView

LinearLayout
(horizontal orientation)

Button

Widgets in the View Hierarchy

- A **widget** is a user interface component, i.e. a building block of a layout.
 - Widgets can show text or graphics, interact with the user, or arrange other widgets on screen
- The widgets of each layout form a hierarchy of **View** objects (“view hierarchy”)
- Elements of the view hierarchy can be
 - visible UI elements
 - e.g. **TextViews**, **Buttons** etc.
 - **ViewGroups** (i.e. widgets containing and arranging other widgets in particular ways)
 - e.g. **LinearLayout**, **FrameLayout**, **TableLayout**, **RelativeLayout**
 - e.g. **LinearLayout** places the contained widgets vertically or horizontally next to each other
- Attributes of the XML elements indicate how the widgets shall be configured.

Widget Attributes

- **layout_width, layout_height:** Widget size
 - **match_parent:** as big as parent widget
 - **wrap_content:** as big as required by child widgets
- **padding:** Space added around widget
 - unit **dp:** density-independent pixels

The screenshot shows the Android Studio interface with the XML layout editor and preview pane. The XML code in the editor defines a vertical LinearLayout containing a TextView and a horizontal LinearLayout. The horizontal LinearLayout contains two buttons, one labeled "true" and the other "false". The preview pane shows a dark-themed mobile application interface with the text "@string/question_text" above the buttons. The buttons are labeled "@STRING/TRUE_BUTTON" and "@STRING/FALSE_BUTTON". The XML code is as follows:

```
activity_main.xml
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

    <TextView
        android:text="@string/question_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp"/>

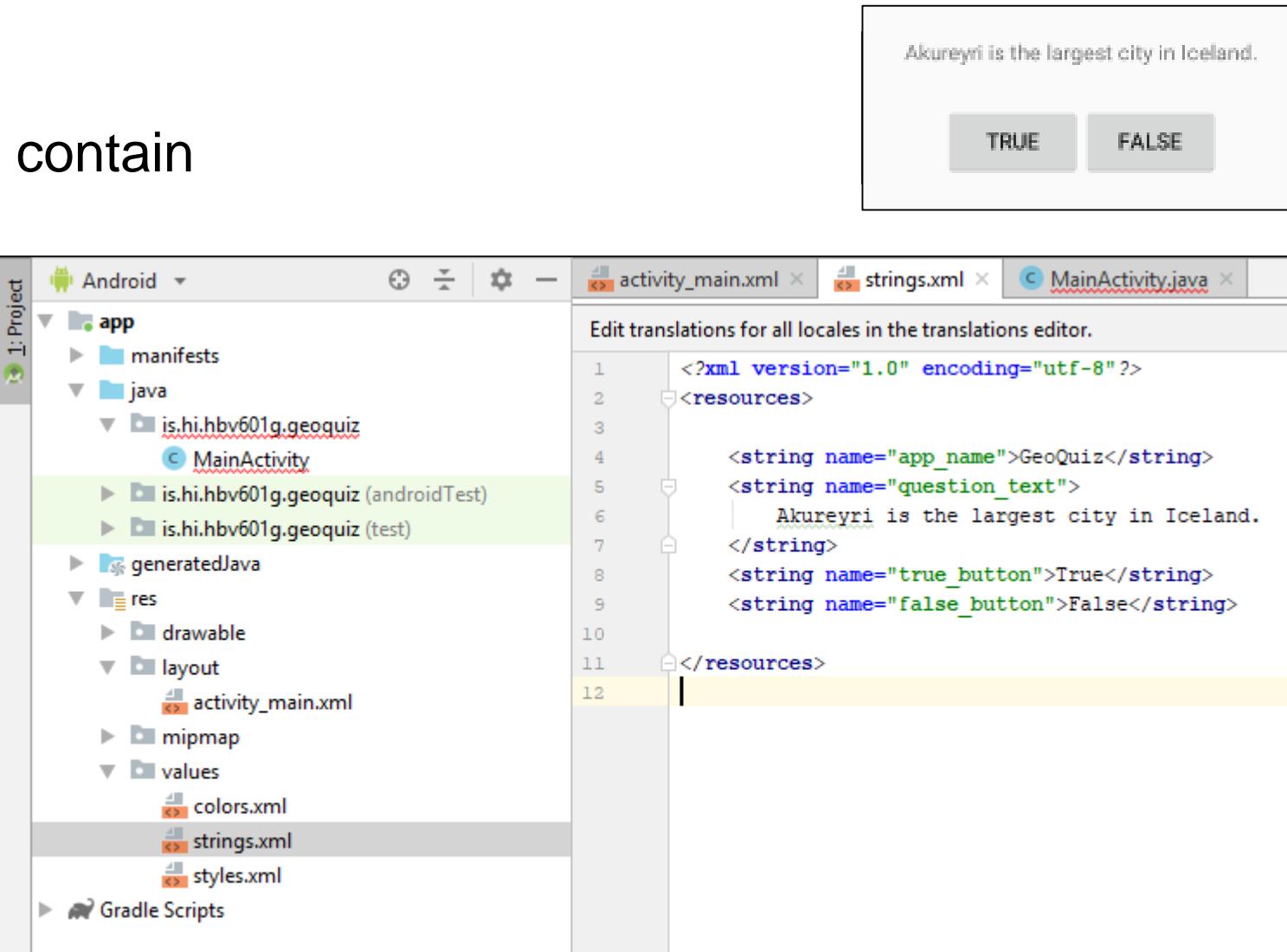
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:id="@+id/true_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/true_button"/>

        <Button
            android:id="@+id/false_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/false_button"/>
    </LinearLayout>
</LinearLayout>
```

Defining String Resources

- **text** attributes of widgets can contain
 - literal text, e.g. "Text"
 - better: References to string resources
 - Format: @**string/name**
- Strings are defined in a strings file
 - Default location: **app/src/main/res/values/strings.xml**
 - Each string is identified by **name** in a **string** element



Akureyri is the largest city in Iceland.

TRUE FALSE

The screenshot shows an Android Studio project structure. The Project tool window on the left displays the app module with its subfolders: manifests, java, res, and Gradle Scripts. The java folder contains a package named is.hi.hbv601g.geoquiz with a MainActivity.java file. The res folder contains drawable, layout, mipmap, and values subfolders. The values folder contains three files: colors.xml, strings.xml (which is currently selected), and styles.xml. The strings.xml file is open in the Editor tool window on the right. It contains XML code defining string resources:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">GeoQuiz</string>
    <string name="question_text">
        Akureyri is the largest city in Iceland.
    </string>
    <string name="true_button">True</string>
    <string name="false_button">False</string>
</resources>
```

Referring to a Layout From an Activity

- When an instance of an activity is created, its **onCreate** method is called
 - To indicate which user interface the activity shall manage, we pass the ID of the desired layout to the **setContentView** method
 - here: **setContentView(R.layout.activity_main)**

- This “inflates” the layout
 - i.e. parses the XML layout file and creates according Java objects for all widgets

The screenshot shows the Android Studio interface with the following details:

- Project Bar:** Shows "1 Project" and the "Android" tab.
- Project Tree:** Under "app", there are "manifests", "java", "is.hi.hbv601g.geoquiz" (containing "MainActivity"), "generatedJava", "res" (with "drawable", "layout", "mipmap", and "values" subfolders), and "strings.xml" files in "values".
- Main Activity Code:** The code for `MainActivity.java` is displayed in the editor:

```
1 package is.hi.hbv601g.geoquiz;
2
3 import android.support.v7.app.AppCompatActivity;
4 import android.os.Bundle;
5
6 public class MainActivity extends AppCompatActivity {
7
8     @Override
9     protected void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.activity_main);
12     }
13 }
14 
```
- Toolbars:** Standard Android Studio toolbars for file operations.

Resources

- A resource is any piece of an app that is not code
 - e.g. images, XML files etc.
- Resources are stored in subdirectories of app/src/main/res, e.g.
 - .../layout/activity_main.xml
 - .../values/strings.xml
- In Java code, resources are referenced using constants defined in the automatically-generated Java class R

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY. */

package is.hi.hbv601g.geoquiz;

public final class R {

    public static final class layout {
        public static final int activity_main=0x7f040019;
        // ...
    }

    public static final class string {
        public static final int false_button=0x7f0a0012;
        public static final int true_button=0x7f0a0015;
        // ...
    }

    public static final class id { ... }
    // ...
}
```

Note: inner class

In-Class Quiz #3: Basic Android Components



- Fill the blanks in the following sentences with the words (A)ctivity, (L)ayout, (R)eource and (W)idget:

- A _____ manages the user's interaction with the UI defined by a _____.
- A _____ defines a set of _____ and their screen positions.
- A view hierarchy contains the _____ of one _____.
- The `setContentView` method of a _____ inflates the given _____.
- Inflating a _____ means creating Java objects for all _____ declared in it.
- A _____ is any piece of an app that is not executable code.
- The auto-generated `R` class contains references to all _____ and _____ defined in XML files under `app/src/main/res/`.

Creating IDs for Widgets

- To add behavior to our buttons, we need to refer to them in Java
 - Define IDs for them by adding `id` attributes to the XML layout file

The `+` sign in `@+id` indicates we are *defining* the ID

`@string` (without `+`) indicates we are *referencing* the ID

The screenshot shows the Android Studio interface. On the left, the XML layout file `activity_main.xml` is displayed:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

    <TextView
        android:text="@string/question_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp"/>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <Button
            android:id="@+id/true_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/true_button"/>

        <Button
            android:id="@+id/false_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/false_button"/>

    </LinearLayout>
</LinearLayout>
```

The preview pane on the right shows a simple UI with a TextView containing the text "Akureyri is the largest city in Iceland." Below it are two buttons labeled "TRUE" and "FALSE".

Referring to Widgets from an Activity

- Recap:

- Defining the Button widgets with the attributes `android:id="@+id/name"` in the layout file created IDs for the buttons in the `R.id` class
- `setContentView` inflates the layout, i.e. parses the XML layout file and creates according Java objects for all widgets
- Both processes (generating the `R` class and generating the widget classes) are automatic!

- To obtain the generated widget classes, pass ID of desired widget to the `findViewById` method and cast the returned `View` to expected type
 - e.g. `mTrueButton = (Button) findViewById(R.id.true_button);`
 - Note: Android naming convention: Class attributes (“member variables”) are prefixed with `m`

```
package is.hi.hbv601g.geoquiz;

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

public class MainActivity extends AppCompatActivity {

    private Button mTrueButton;
    private Button mFalseButton;

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

        mTrueButton = (Button) findViewById(R.id.true_button);
        mFalseButton = (Button) findViewById(R.id.false_button);
    }
}
```



Setting Listeners

- Android apps are event-driven
 - i.e. after starting, they wait (“listen”) for user- or system-initiated events
- Android provides listener interfaces for various events, e.g.
 - **View.OnClickListener** for short clicks/touches on widgets
 - **View.OnLongClickListener** for long clicks/touches on widgets
- You provide the listener implementation
 - i.e. code saying what happens upon the event

```
package is.hi.hbv601g.geoquiz;

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

public class MainActivity extends AppCompatActivity {

    private Button mTrueButton;
    private Button mFalseButton;

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

        mTrueButton = (Button) findViewById(R.id.true_button);
        mTrueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                // to be implemented
            }
        });

        mFalseButton = (Button) findViewById(R.id.false_button);
    }
}
```

Note: Anonymous inner class



Anonymous Inner Classes

- Example: `setOnClickListener` expects an `OnClickListener` instance
 - We could write an `OnClickListener` implementation in a regular Java class (e.g. `TrueButtonOnClickListener.java`) as a separate file somewhere in our project...
 - ...store an instance of it in a local variable, such as
`trueButtonOnClickListener = new TrueButtonOnClickListener()`
 - ...and then pass this instance to `setOnClickListener`, i.e.
`mTrueButton.setOnClickListener(trueButtonOnClickListener)`
 - But it's more compact to do all that (implementation, instantiation, param passing) at once:

```
mTrueButton.setOnClickListener(new View.OnClickListener() {  
    @Override  
    public void onClick(View v) { ... }  
});
```

- Advantages
 - Avoiding lots of small listener class files cluttering up our project
 - Maintaining the listener implementation exactly where it is needed
 - “when this event happens to this widget, this code will be executed”

It's OK for the class to be anonymous since we are referring to it only here

Making Toasts

- A **toast** is a short message display that does not require any user interaction.
- To create a toast, call the following method from the **Toast** class:

```
public static Toast.makeText(Context context, int resId, int duration)
```

- **Context** parameter usually provides reference to current activity
 - **resId** refers to the string that shall be displayed (needs to be defined in **strings.xml**)
 - **duration** indicates how long to show toast (**Toast.LENGTH_SHORT** or **_LONG**)
- To show the toast, call its **show** method

```
mTrueButton = (Button) findViewById(R.id.true_button);
mTrueButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Toast.makeText(MainActivity.this, R.string.correct_toast, Toast.LENGTH_SHORT).show();
    }
});
```

We cannot use just **this**, since that would refer to the anonymous class!

makeText is a static factory method returning a **Toast** instance that we now show

Complete Activity with Event Listeners

```
public class MainActivity extends AppCompatActivity {

    private Button mTrueButton;
    private Button mFalseButton;

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

        mTrueButton = (Button) findViewById(R.id.true_button);
        mTrueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(MainActivity.this, R.string.correct_toast, Toast.LENGTH_SHORT).show();
            }
        });

        mFalseButton = (Button) findViewById(R.id.false_button);
        mFalseButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(MainActivity.this, R.string.incorrect_toast, Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

/java/is/hi/hbv601g/geoquiz/MainActivity.java



Recap: Corresponding Layout and String Resource File

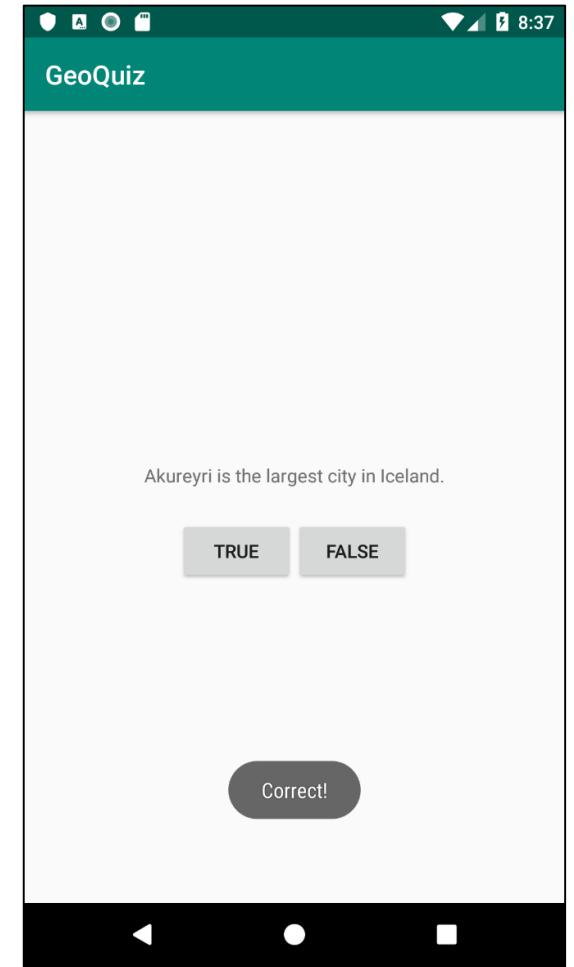
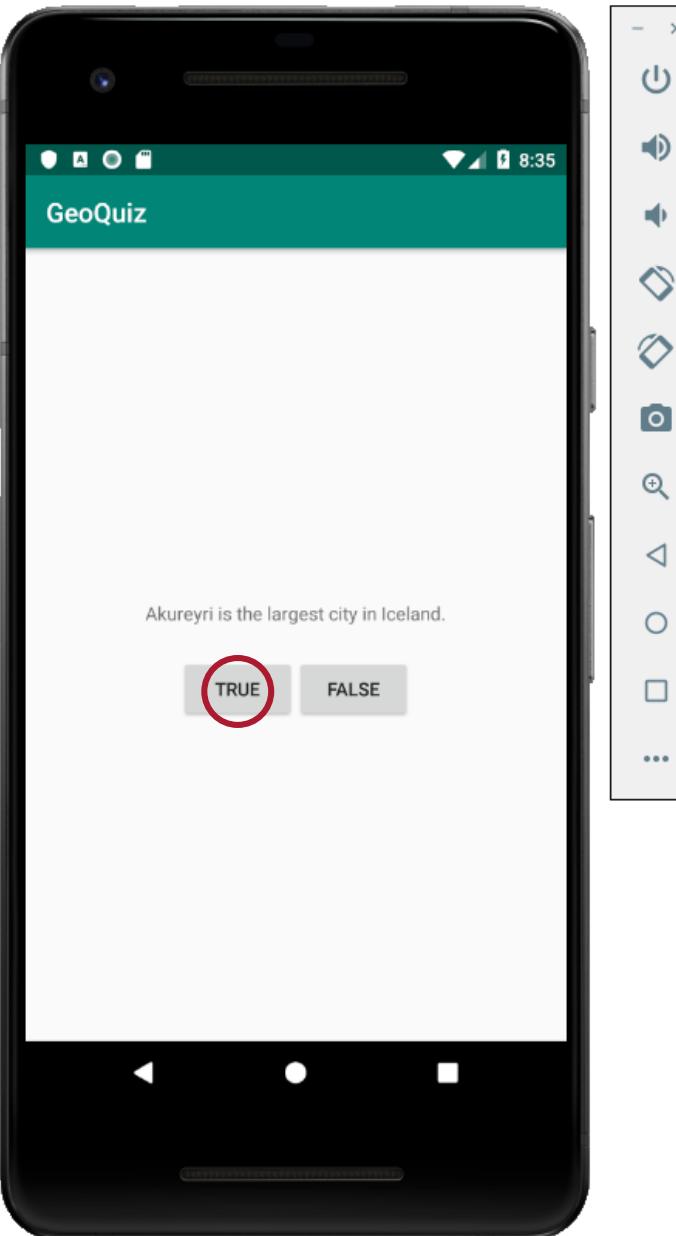
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:gravity="center"  
    android:orientation="vertical">  
  
    <TextView  
        android:text="@string/question_text"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:padding="24dp"/>  
  
    <LinearLayout  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:orientation="horizontal">  
  
        <Button  
            android:id="@+id/true_button"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="@string/true_button"/>  
  
        <Button  
            android:id="@+id/false_button"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="@string/false_button"/>  
  
    </LinearLayout>  
  
</LinearLayout>
```

/res/layout/activity_main.xml

/res/values/strings.xml

```
<?xml version="1.0" encoding="utf-8" ?>  
<resources>  
  
    <string name="app_name">GeoQuiz</string>  
    <string name="question_text">  
        Akureyri is the largest city in Iceland.  
    </string>  
    <string name="true_button">True</string>  
    <string name="false_button">False</string>  
    <string name="correct_toast">Correct!</string>  
    <string name="incorrect_toast">Incorrect!</string>  
  
</resources>
```

Resulting Application



- ...obviously requires some more logic 😊



Hugbúnaðarverkefni 2 / Software Project 2

6. Android Control Flow

HBV601G – Spring 2020

Matthias Book



HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRVÍSINDASVIÐ
ÍÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD

Reboot Hack

15-16 February, Askja



Re:boot Hack

Friday 14th February 2020

- 18:00 Meet & greet - participants
Askja

Saturday 15th February 2020

- 8:00 Registration opens
Askja entrance
- 8:30 Breakfast & schedule presented
Askja
- 10:00 Opening ceremony
Askja, room T32
- 10:30 Team forming
Askja
- 11:30 Lunch & hacking begins
Askja
- 13:00 Workshops start
Rooms T29 & T30
- 15:00 Coffee break
Askja
- 17:00 Workshops end
- 19:00 Dinner
Askja
- 21:00 Pitch workshop
- 21:50 Yoga
- 22:30 Surprise
- 23:30 Midnight snack provided by Ölgerðin

Sunday 16th February 2020

- 8:00 Breakfast
- 11:00 Submissions close
- 11:30 Lunch
- 12:00 Pitches
Askja
- 13:30 Pitches end & judges deliberate
- 14:15 Top 5 teams present & award ceremony
Askja, room T32

  RebootHackIceland #reboothack



Leikjavæðing
sparnaðar



Hvernig aukum við
starfsánægju með
tækni?



Traust í viðskiptum
framtíðarinnar



Hvernig getur samspli
lista, hönnunar og
tækni spornað við
loftlagshamförum?



Hvernig verður
tryggingaþjónust
a í framtíðinni?



Hvernig getur tækni
gert sölu og
framleiðslu
Ölgerðarinnar
sjálfbærari?



Hver er framtíð starfa án
staðsetninga?



Hvernig gerum við
starfsfólk meðvitað um
hættu skipulagðra
netglæpahópa?

Sign up at reboothack.is

1st prize: 100.000 kr.



HÁSKÓLI ÍSLANDS

Miðmisseriskönnun

Evaluate this course on Uglá!
(open until Sat 15 Feb)



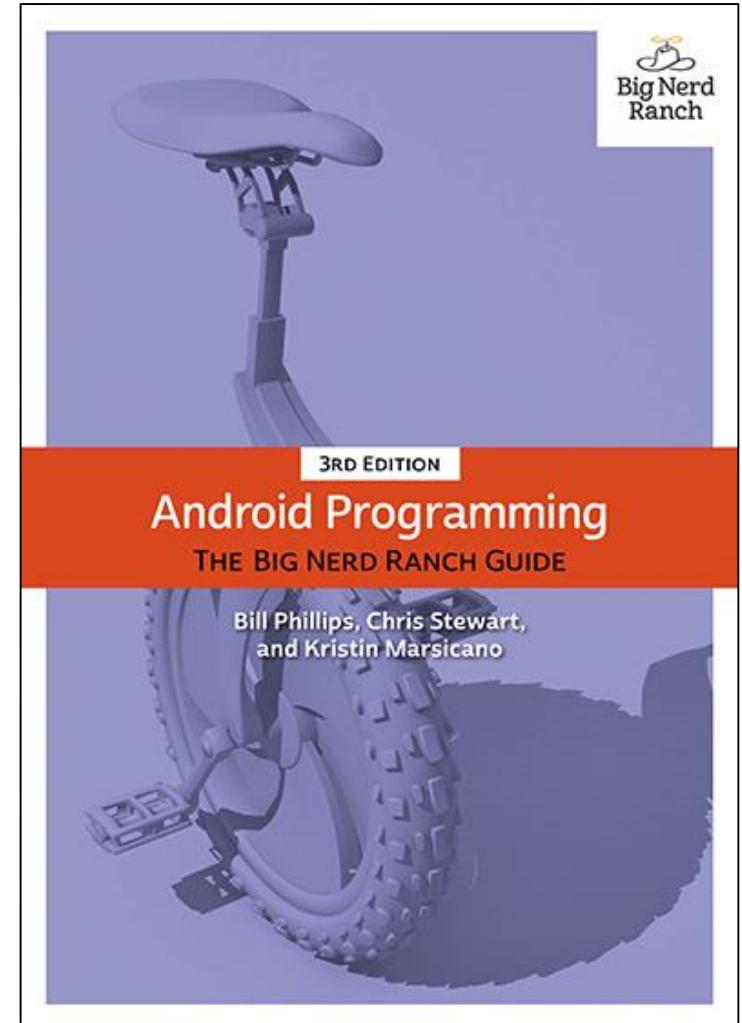
Server-Side Redesign Advice

- How to turn your Spring web application from HBV501G into a RESTful back-end for your HBV601G Android app
- Code-along demo in next week's consultation
 - Thu 20 Feb 13:20-14:20 with Kristján Pétur in VRII-152



Examples and Template Projects

- Most lecture examples are adapted from the book
 - Bill Phillips, Chris Stewart, Kristin Marsicano: Android Programming, 3rd Edition. Big Nerd Ranch, 2016
- Source code of all examples in the book
 - available as ZIP archive in the Verkefni folder in Uglar
- You could use the code examples from the following chapters as templates for your app:
 - Chapter 6 – working with activities, layouts and intents
 - Chapter 17 – working with fragments, complex views, DBs...
 - Chapter 22 – working with multimedia assets
 - Chapter 30 – working with web services
 - Chapter 31ff – working with gestures, GPS, animations etc.



In-Class Quiz #4 Prep

- Please prepare a small scrap of paper with the following information:

ID: _____ @hi.is Date: _____

a) _____ e) _____ i) _____
b) _____ f) _____ j) _____
c) _____ g) _____ k) _____
d) _____ h) _____

- During class, I'll show you questions that you can answer very briefly
 - Just numbers or letters, no elaboration
- Hand in your scrap at the end of class
- All questions in a quiz weigh same
- All quizzes (ca. 10-12 throughout semester) have the same weight
 - Your worst 2 quizzes will be disregarded
- Overall quiz grade counts as optional question worth 7.5% on final exam



In-Class Quiz 3 Solution



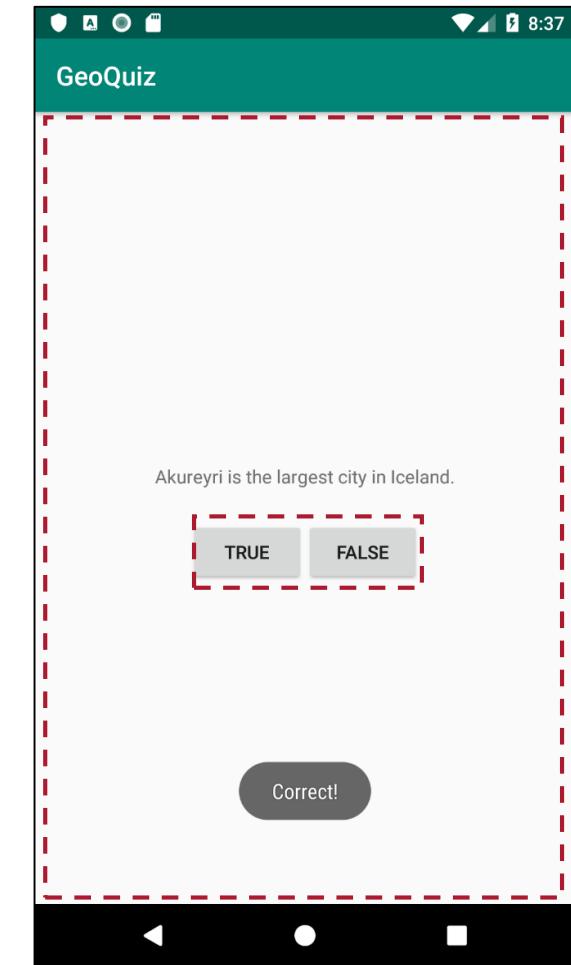
- Fill the blanks in the following sentences with the words (A)ctivity, (L)ayout, (R)eource and (W)idget:

- An **activity** manages the user's interaction with the UI defined by a **layout**.
- A **layout** defines a set of **widgets** and their screen positions.
- A view hierarchy contains the **widgets** of one **layout**.
- The **setContentView** method of an **activity** inflates the given **layout**.
- Inflating a **layout** means creating Java objects for all **widgets** declared in it.
- A **resource** is any piece of an app that is not executable code.
- The auto-generated **R** class contains references to all **resources** and **widgets** defined in XML files under **app/res/**.

Recap: Layout Implementation

- A **layout** defines a set of UI widgets and their screen positions.
- **Widgets** can show text or graphics, interact with the user, or arrange other widgets on screen
- The widgets of each layout form a hierarchy of View objects (the “**View hierarchy**”)

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    xmlns:tools="http://schemas.android.com/tools"  
    android:layout_width="match_parent"  
    android:layout_height="match_parent"  
    android:gravity="center"  
    android:orientation="vertical">  
  
    <TextView  
        android:text="@string/question_text"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:padding="24dp"/>  
  
    <LinearLayout  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:orientation="horizontal">  
  
        <Button  
            android:id="@+id/true_button"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="@string/true_button"/>  
  
        <Button  
            android:id="@+id/false_button"  
            android:layout_width="wrap_content"  
            android:layout_height="wrap_content"  
            android:text="@string/false_button"/>  
  
    </LinearLayout>  
    </LinearLayout>
```



Recap: Activity Implementation

- An **activity** is responsible for managing user interaction with a screen of information
 - i.e. it implements a part of the functionality of your app
- Android apps are event-driven
 - i.e. after starting, they wait (“listen”) for a user- or system-initiated event to occur
- Android provides listener interfaces for many events
 - You override the listener with code determining what happens upon the event

```
public class MainActivity extends AppCompatActivity {  
  
    private Button mTrueButton;  
    private Button mFalseButton;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        mTrueButton = (Button) findViewById(R.id.true_button);  
        mTrueButton.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                Toast.makeText(MainActivity.this,  
                    R.string.correct_toast, Toast.LENGTH_SHORT).show();  
            }  
        });  
  
        mFalseButton = (Button) findViewById(R.id.false_button);  
        mFalseButton.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                Toast.makeText(MainActivity.this,  
                    R.string.incorrect_toast, Toast.LENGTH_SHORT).show();  
            }  
        });  
    }  
}
```

Event listener in anonymous inner class

/java/is/hi/hbv601g/geoquiz/MainActivity.java



Recap: Resources

- A resource is any piece of an app that is not source code
 - e.g. images, XML files etc.
- Resources are stored in subdirectories of app/src/main/res, e.g.
 - .../layout/activity_main.xml
 - .../values/strings.xml
- In source code, resources are referenced using constants defined in the automatically-generated class **R**

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">GeoQuiz</string>
    <string name="question_text">
        Akureyri is the largest city in Iceland.
    </string>
    <string name="true_button">True</string>
    <string name="false_button">False</string>
    <string name="correct_toast">Correct!</string>
    <string name="incorrect_toast">Incorrect!</string>

</resources>
```

```
/* AUTO-GENERATED FILE.  DO NOT MODIFY. */

package is.hi.hbv601g.geoquiz;

public final class R {

    public static final class layout {
        public static final int activity_main=0x7f040019;
        // ...
    }

    public static final class string {
        public static final int false_button=0x7f0a0012;
        public static final int true_button=0x7f0a0015;
        // ...
    }

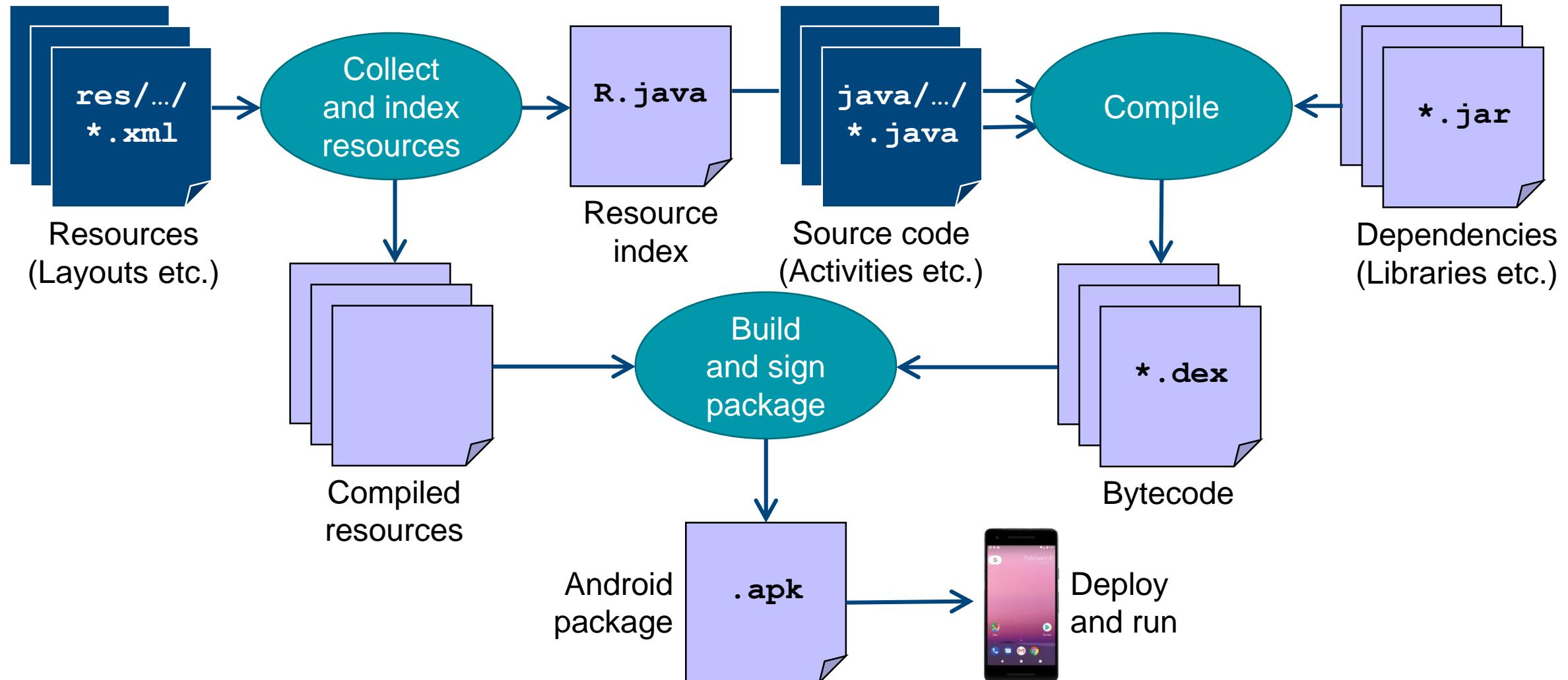
    public static final class id { ... }

    // ...
}
```

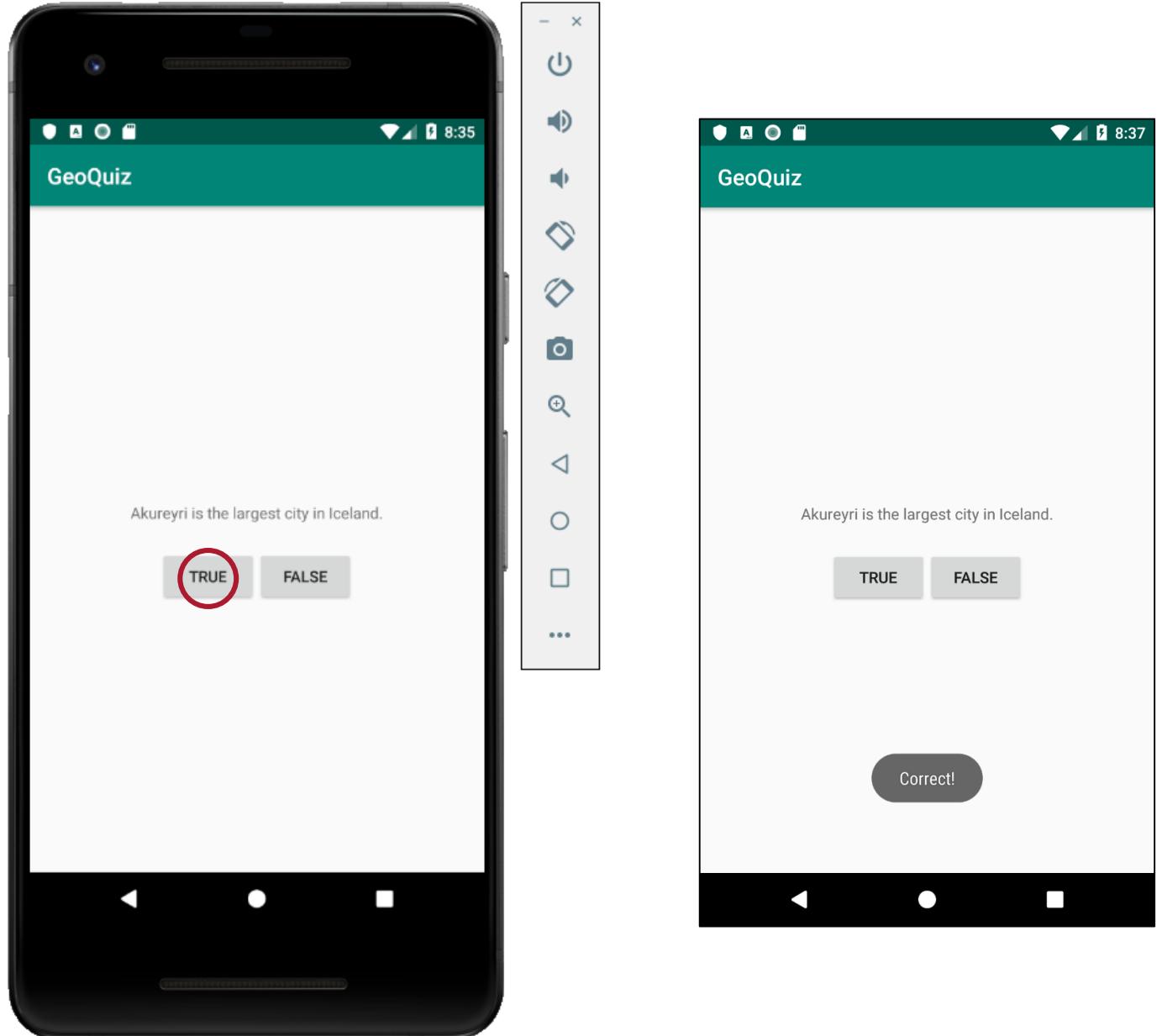
/res/values/
strings.xml



Under the Hood: The Android Build Process (simplified)



Recap: Running Application



- ...obviously requires some more logic 😊

The Model-View-Controller Pattern in Android Apps

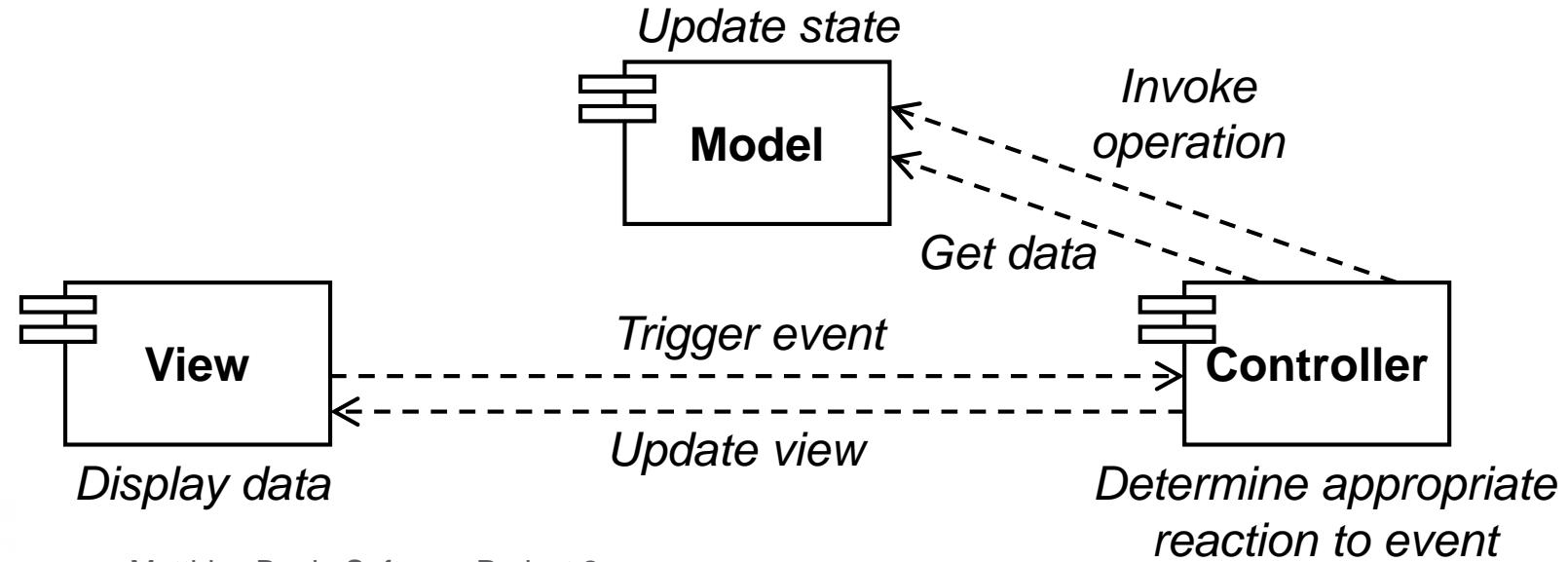
see also:

- Phillips et al.: Android Development, Ch. 2



Model-View-Controller (MVC) Pattern in Android Apps

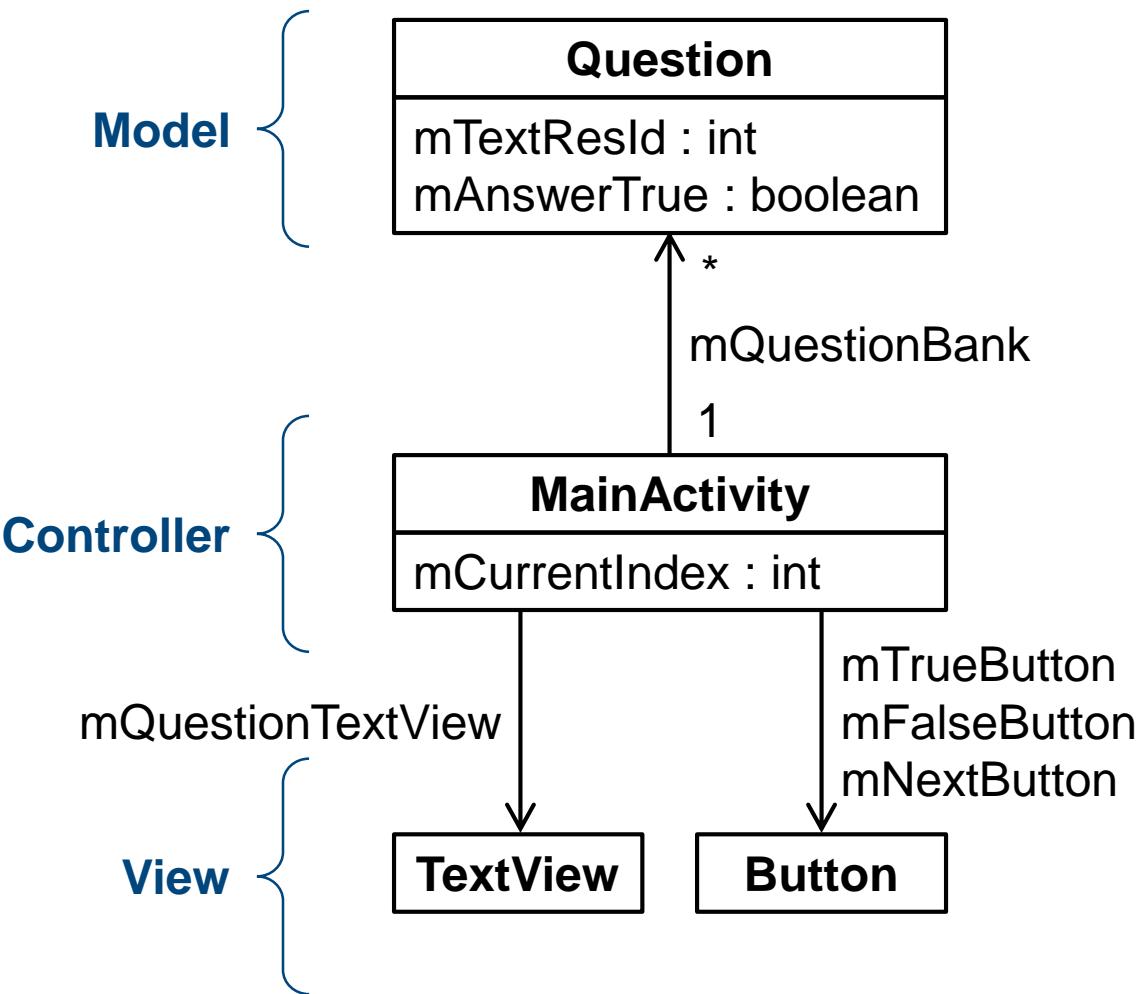
- **Model:** Objects responsible for holding the app's data and business logic
- **View:** Objects that know how to draw themselves on the screen and how to respond to user input
- **Controller:** Objects that respond to events triggered by views, invoke logic on the model, and thus manage the data and control flow between model and view



Example: Managing Questions in the GeoQuiz App

- Instead of just having one static question with a hard-wired answer, the app should let users go through a series of questions.

- We need:
- in the Model layer:**
Objects storing questions and answers
 - in the View layer:**
Buttons for navigating between questions
 - in the Controller layer:**
Logic that ties buttons to questions, and creates toasts according to the answers



Model: Question.java

POJO describing a question
and its correct answer

Note: We use **int** instead
of **String** since we won't
store the literal question
strings here, but
references to the string
resource file (to enable
easier internationalization)

```
package is.hi.hbv601g.geoquiz;

public class Question {
    private int mTextResId;
    private boolean mAnswerTrue;

    public Question(int textResId, boolean answerTrue) {
        mTextResId = textResId;
        mAnswerTrue = answerTrue;
    }

    public int getTextResId() {
        return mTextResId;
    }

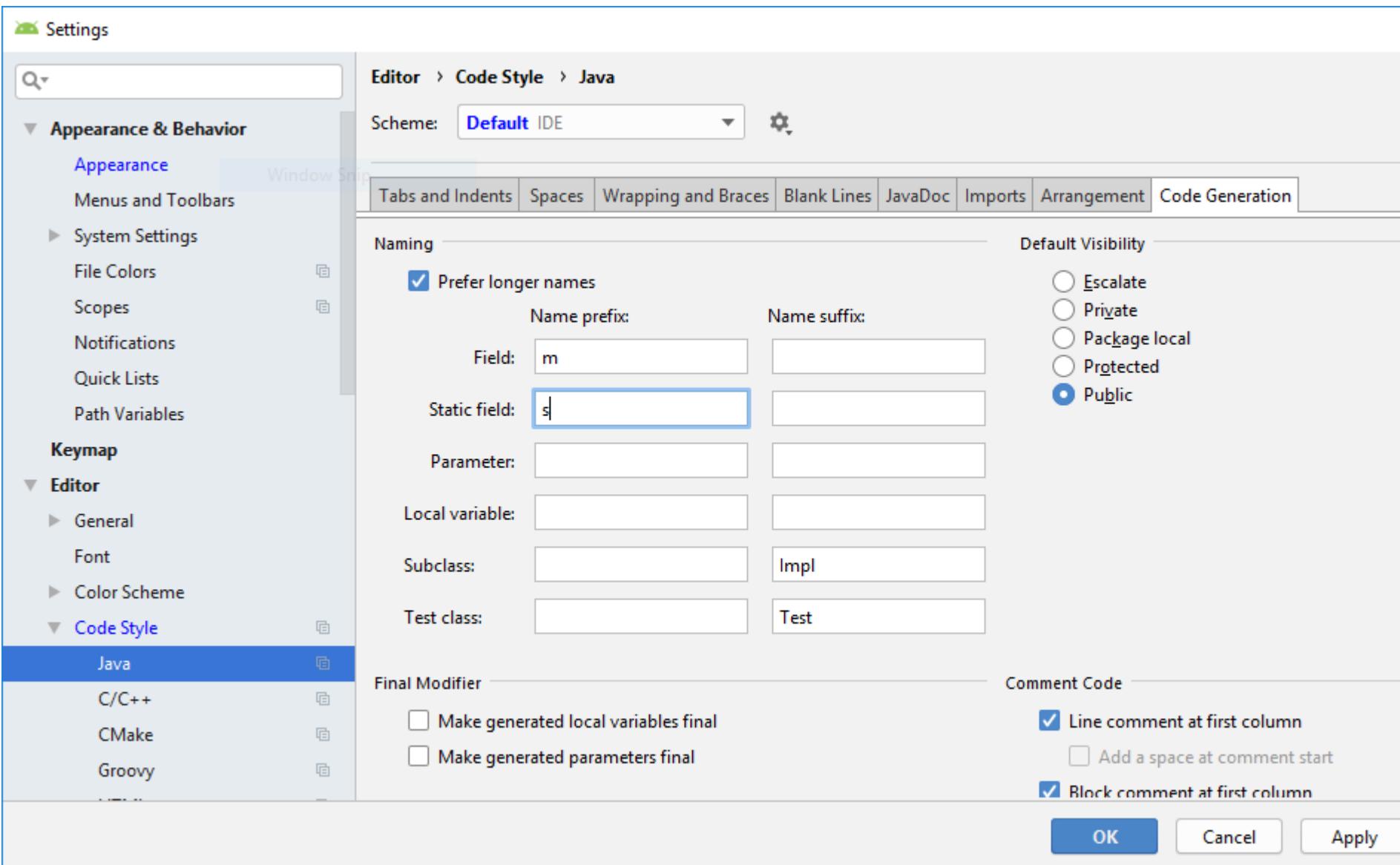
    public void setTextResId(int textResId) {
        mTextResId = textResId;
    }

    public boolean isAnswerTrue() {
        return mAnswerTrue;
    }

    public void setAnswerTrue(boolean answerTrue) {
        mAnswerTrue = answerTrue;
    }
}
```

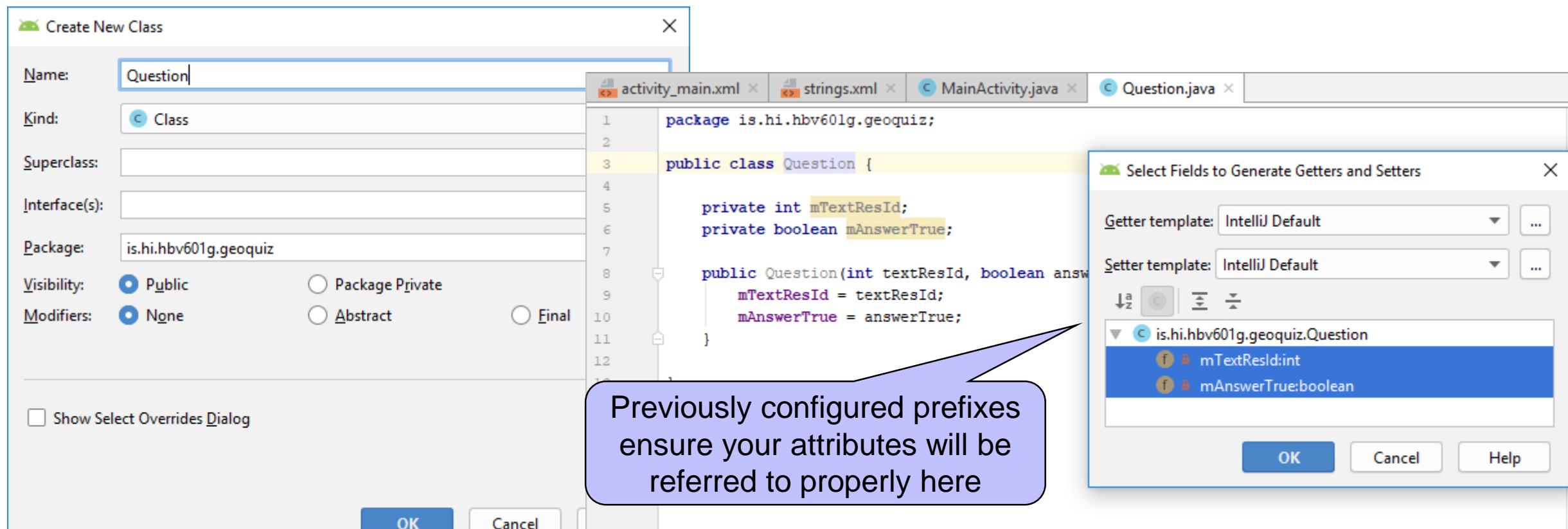
IDE Tip: Prefixes

- In “File > Settings” menu
- Open the “Editor > Code Style > Java” dialog
- In the “Code Generation” tab, enter `m` and `s` prefixes for fields and static fields
 - to enable proper generation of getters and setters



IDE Tip: Adding a New Class; Generating Constructor and Accessors

- Right-click on package where you want the class, select “New > Java Class...”
- After coding the attributes, right-click on class name and select “Generate...”
 - Choose attributes that generated code shall apply to
 - Let IDE generate constructor, getters and setters



View: activity_main.xml

Rather than referring to a static string (`@string/question_text`), the **TextView** for the question now gets its own ID through which we can modify it

```
...  
  
<TextView  
    android:id="@+id/question_text_view"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:padding="24dp"/>  
  
<LinearLayout  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:orientation="horizontal">  
  
<Button  
    android:id="@+id/next_button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/next_button"/>  
  
...  
  
</LinearLayout>  
  
...
```

New “Next” button



View: strings.xml

```
<resources>
    <string name="app_name">GeoQuiz</string>
    <string name="true_button">True</string>
    <string name="false_button">False</string>
    <string name="next_button">Next</string>
    <string name="incorrect_toast">Incorrect!</string>
    <string name="correct_toast">Correct!</string>
    <string name="action_settings">Settings</string>
    <string name="question_oceans">The Pacific Ocean is larger than the Atlantic
        Ocean.</string>
    <string name="question_mideast">The Suez Canal connects the Red Sea and the Indian
        Ocean.</string>
    <string name="question_africa">The source of the Nile River is in Egypt.</string>
    <string name="question_americas">The Amazon River is the longest river in the
        Americas.</string>
    <string name="question_asia">Lake Baikal is the world's oldest and deepest freshwater
        lake.</string>
</resources>
```

Additional strings for Next
button and questions



Controller: MainActivity.java

Setting up the QuestionBank

```
public class MainActivity extends AppCompatActivity {  
  
    private Button mTrueButton;  
    private Button mFalseButton;  
    private Button mNextButton;  
    private TextView mQuestionTextView;  
  
    private Question[] mQuestionBank = new Question[] {  
        new Question(R.string.question_oceans, true),  
        new Question(R.string.question_mideast, false),  
        new Question(R.string.question_africa, false),  
        new Question(R.string.question_americas, true),  
        new Question(R.string.question_asia, true)  
    };  
  
    private int mCurrentIndex = 0;  
  
    // ...
```

Just a simple array for now – we'll see more scalable solutions for data storage later



Controller: MainActivity.java

Enabling Updates of the Displayed Question

```
public class MainActivity extends AppCompatActivity {
    // ...
    private TextView mQuestionTextView;

    private void updateQuestion() {
        int question = mQuestionBank[mcurrentIndex].getTextResId();
        mQuestionTextView.setText(question);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // ...
        mQuestionTextView = (TextView) findViewById(R.id.question_text_view);
        // ...
    }
    // ...
}
```

2. Retrieve text for current question and update the `QuestionTextView` to display it

1. Retrieve the `QuestionTextView` object from the view hierarchy



Controller: MainActivity.java

Handling Clicks on the Next Button

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        // ...

        mNextButton = (Button) findViewById(R.id.next_button);
        mNextButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mCurrentIndex = (mCurrentIndex + 1) % mQuestionBank.length;
                updateQuestion();
            }
        });
        updateQuestion();
    }
    // ...
}
```

After all the setup is complete,
display first question

Increment question index
(modulo total number of
questions) and display
new question



Controller: MainActivity.java

Checking Answer and Displaying Response

```
public class MainActivity extends AppCompatActivity {  
    private void checkAnswer(boolean userPressedTrue) {  
        boolean answerIsTrue = mQuestionBank[mcurrentIndex].isAnswerTrue();  
  
        int messageResId = 0;  
  
        if (userPressedTrue == answerIsTrue) {  
            messageResId = R.string.correct_toast;  
        } else {  
            messageResId = R.string.incorrect_toast;  
        }  
  
        Toast.makeText(this, messageResId, Toast.LENGTH_SHORT).show();  
    }  
    // ...  
}
```

Retrieve correct answer for current question

Determine appropriate response

Create and show toast with response



Controller: MainActivity.java

Connecting the Buttons to the Answer Checking Code

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        mTrueButton = (Button) findViewById(R.id.true_button);
        mTrueButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { checkAnswer(true); }
        });

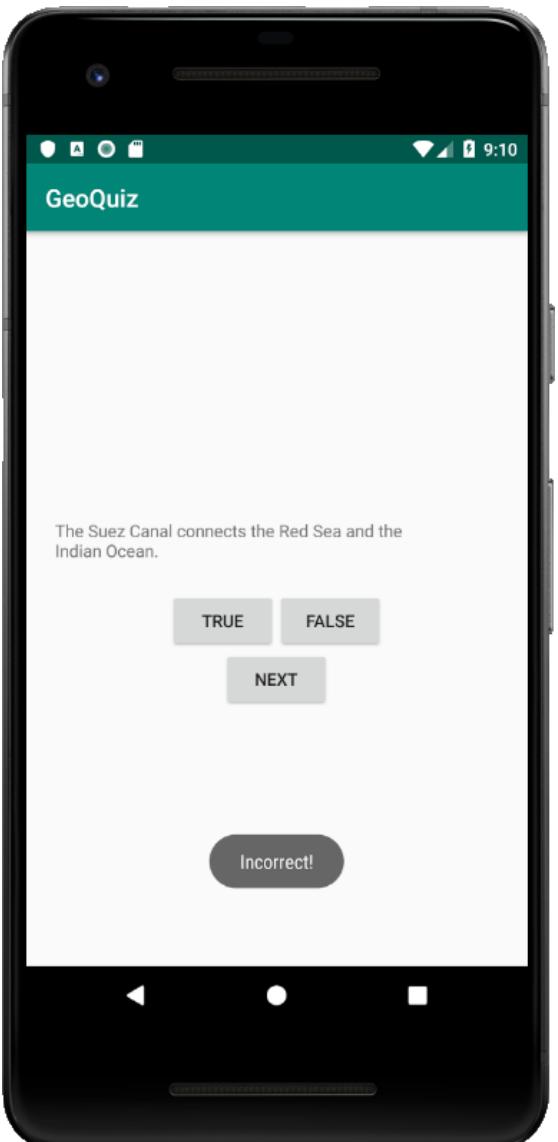
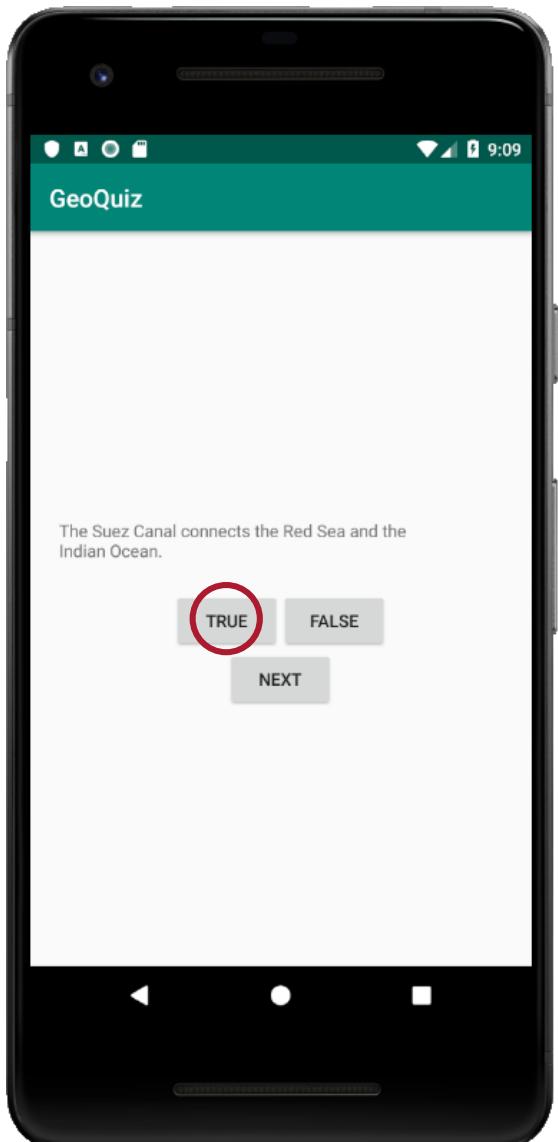
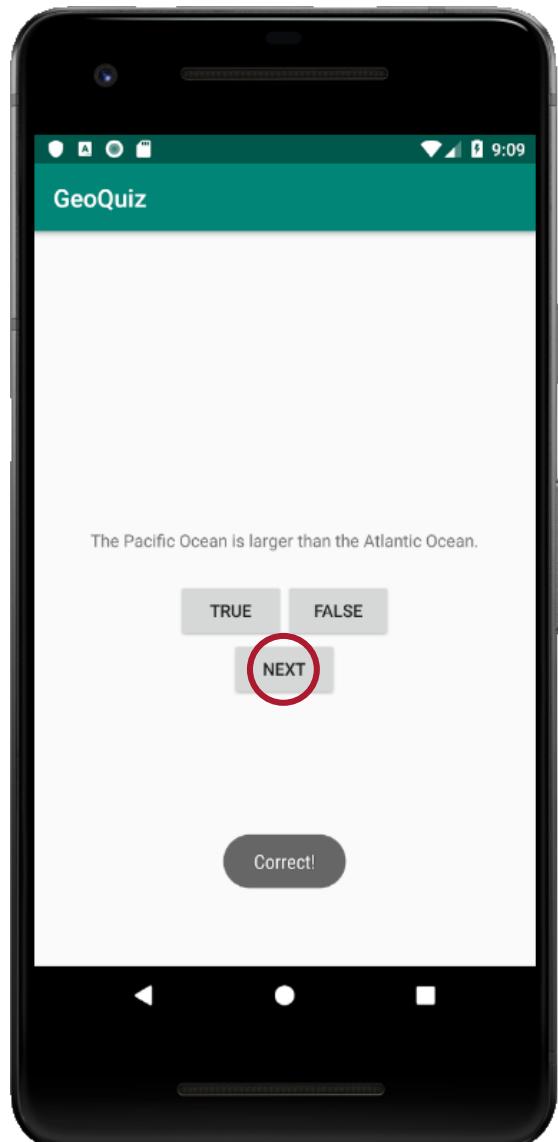
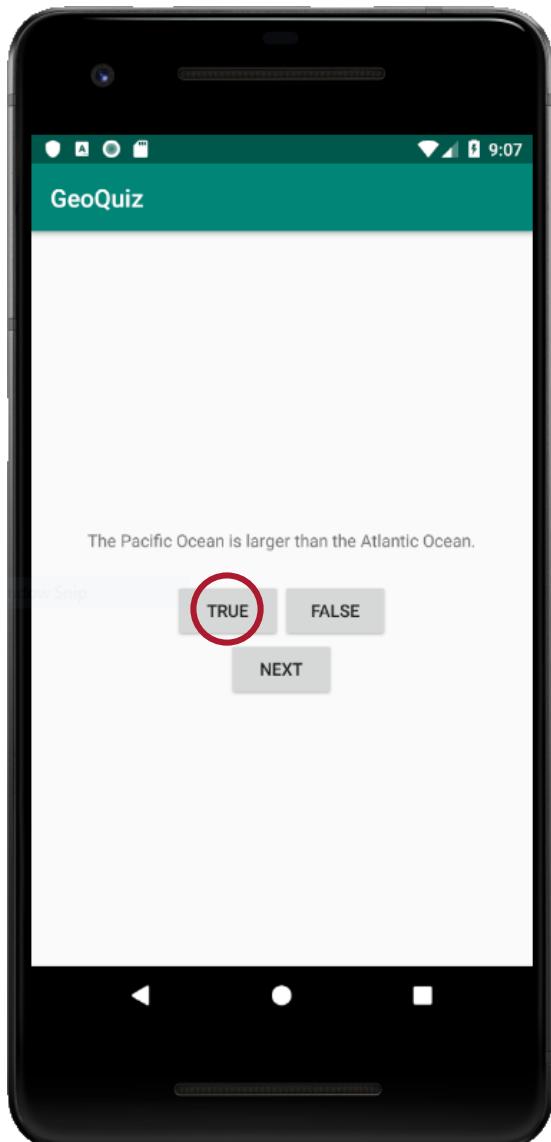
        mFalseButton = (Button) findViewById(R.id.false_button);
        mFalseButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) { checkAnswer(false); }
        });
        // ...
    }
    // ...
}
```

Check if **true** is correct answer

Check if **false** is correct answer



Running the App Again

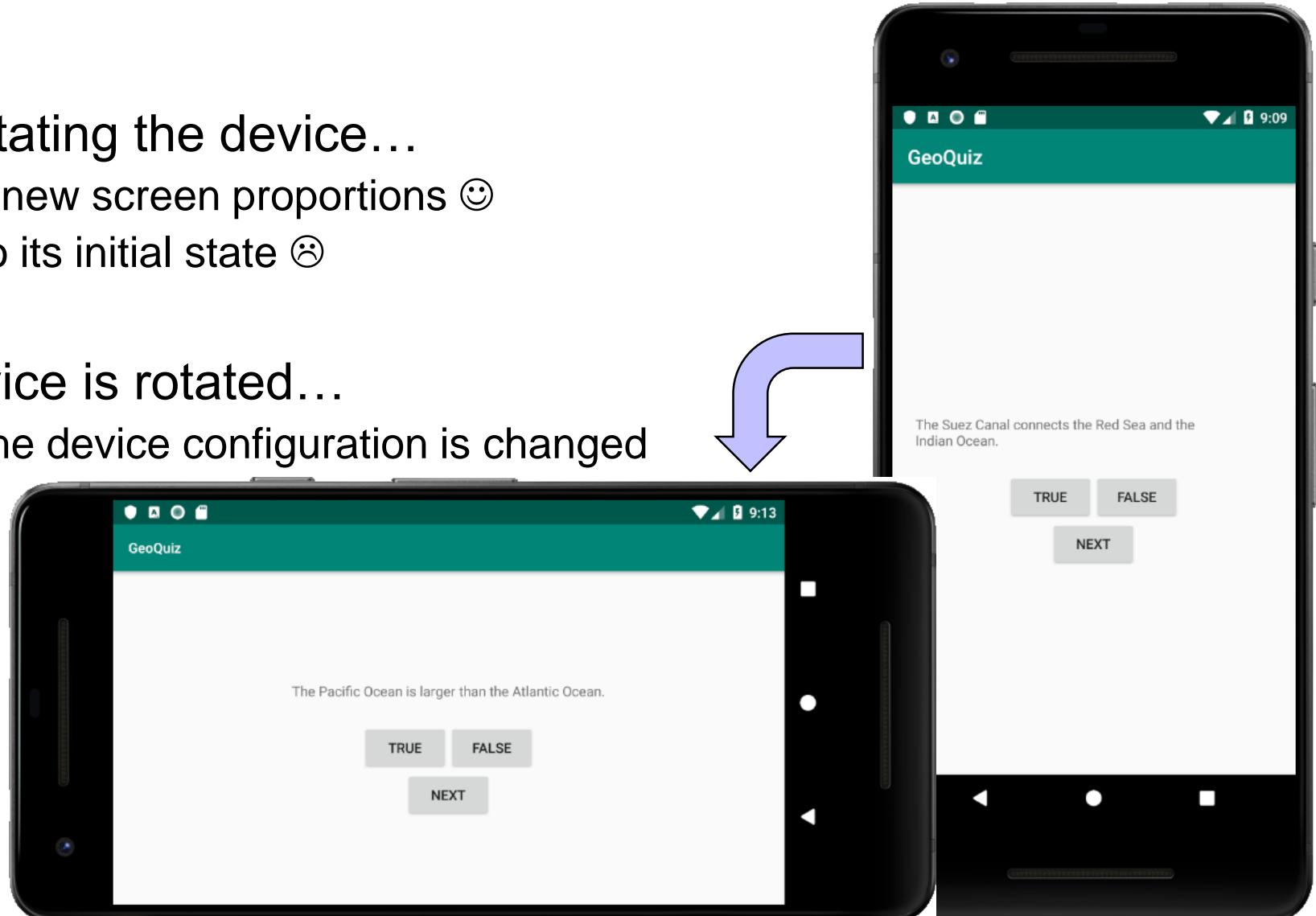


Rotating the Device

- **Observation:** When rotating the device...
 - the layout adapts to the new screen proportions ☺
 - but the activity reverts to its initial state ☹

- **Reason:** When the device is rotated...
 - more generally: When the device configuration is changed

- ...the running activity is destroyed and recreated to match the new configuration
 - i.e. **onCreate** is called again, which resets our question counter



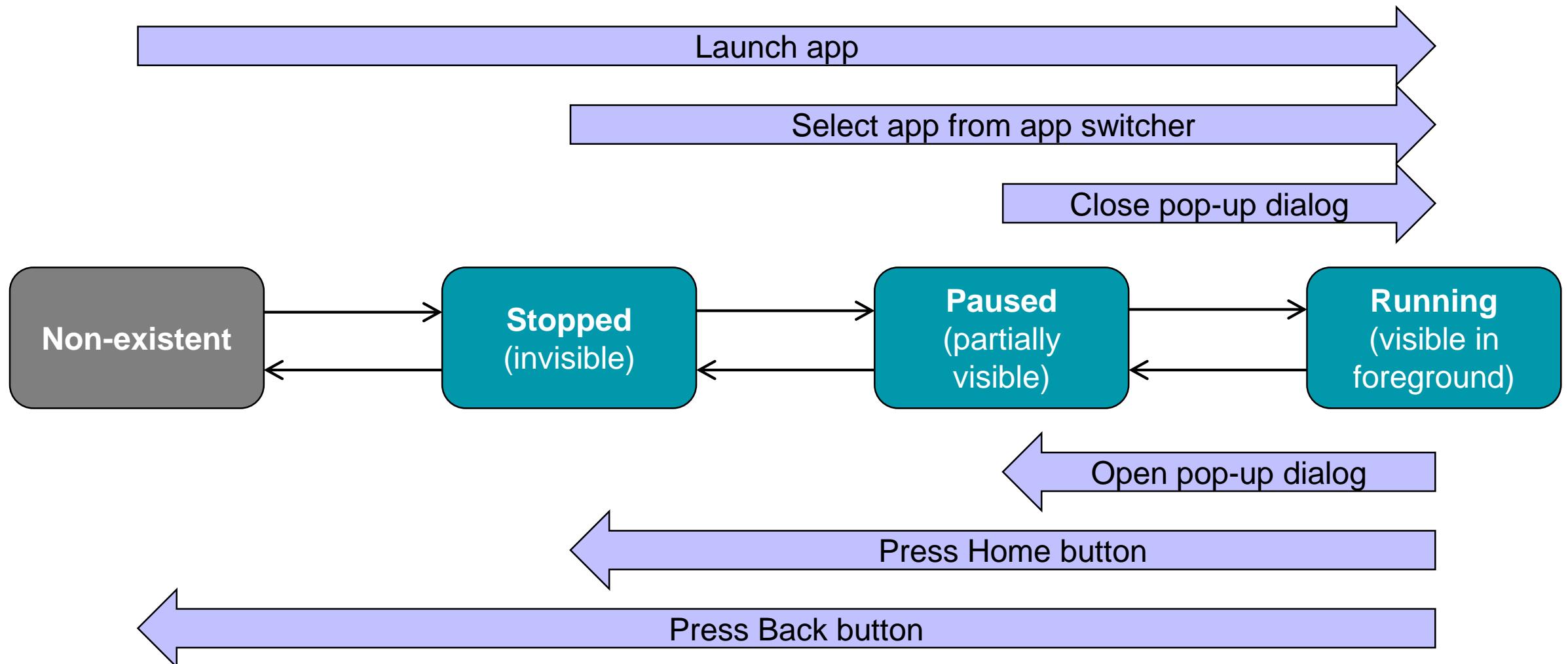
The Activity Lifecycle

see also:

- Phillips et al.: Android Development, Ch. 3
- <http://developer.android.com/guide/components/activities.html>

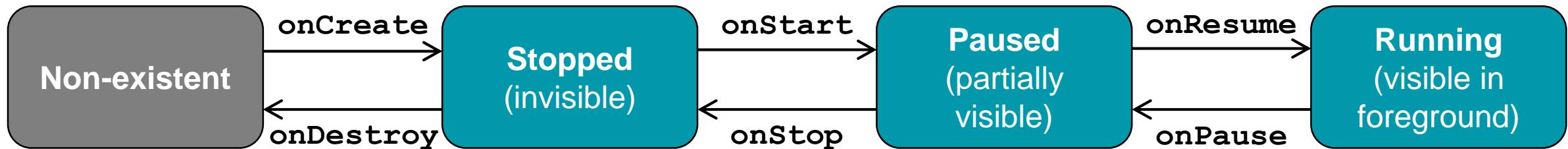


The Activity Lifecycle



The Activity Lifecycle

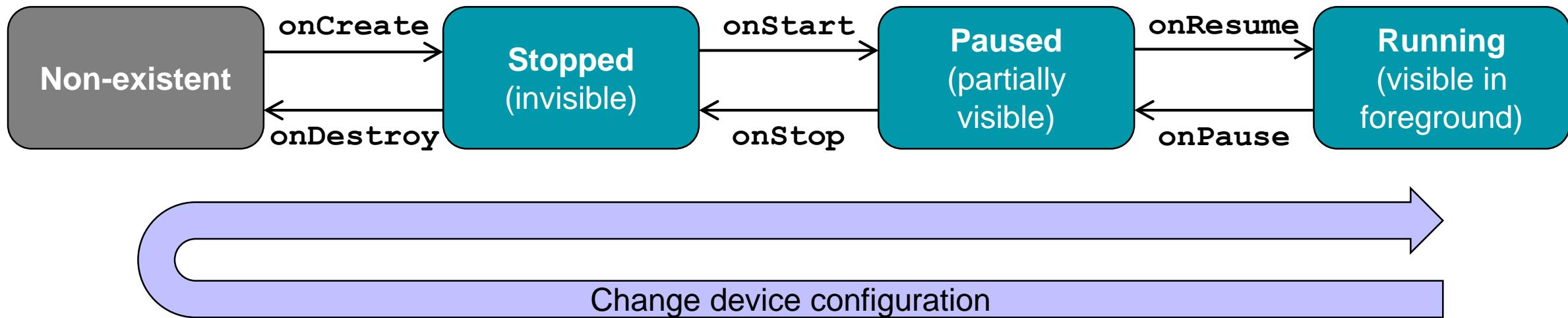
- Activities may transition between three states during their lifecycle
 - e.g. from Non-existent through Stopped and Paused to Running upon launching the app
 - e.g. from Running through Paused to Stopped as a user switches between apps
 - e.g. from Running through Paused and Stopped to Non-existent upon pressing Back button



- Various `on...` methods can be overridden to *react* to these transitions
 - The methods are called by the Android Operating System, *not* the activity!
 - When overriding these, always call superclass' method first (e.g. `super.onPause()`)

Reacting to Device Configuration Changes

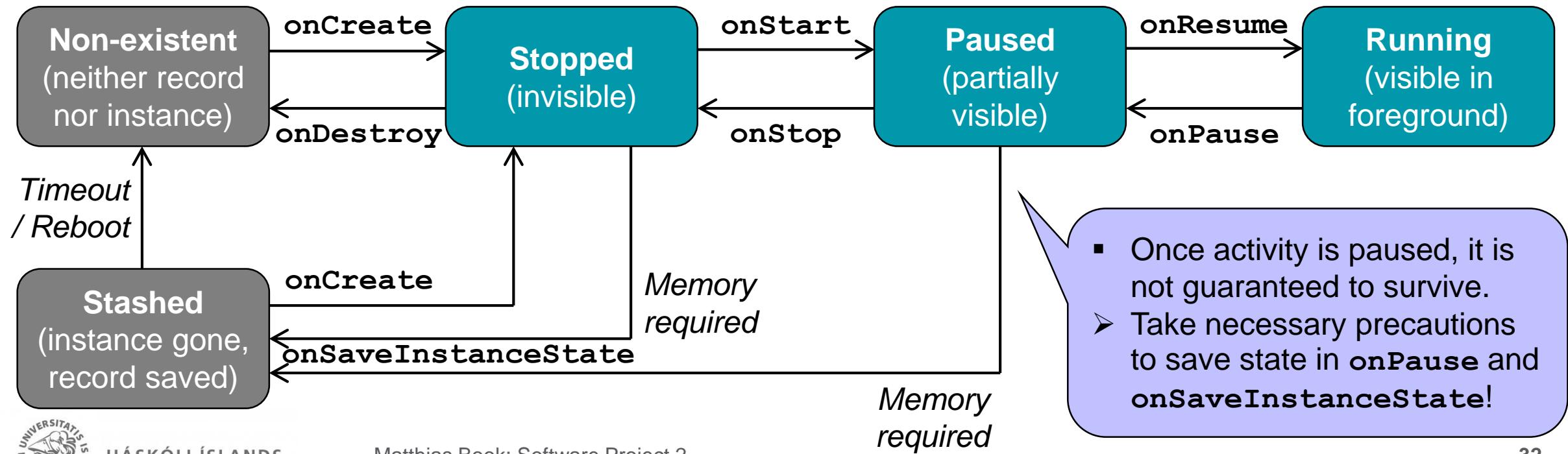
- Some aspects of the device configuration may change at runtime
 - e.g. screen orientation, keyboard type, dock mode, language, etc.
- To work optimally under the changed configuration, an activity may require different resources (e.g. a different layout or different graphics)
 - To re-initialize the resources, the activity depends on **onCreate** being called again



➤ Call order: **onPause**, **onStop**, **onDestroy**, **onCreate**, **onStart**, **onResume**

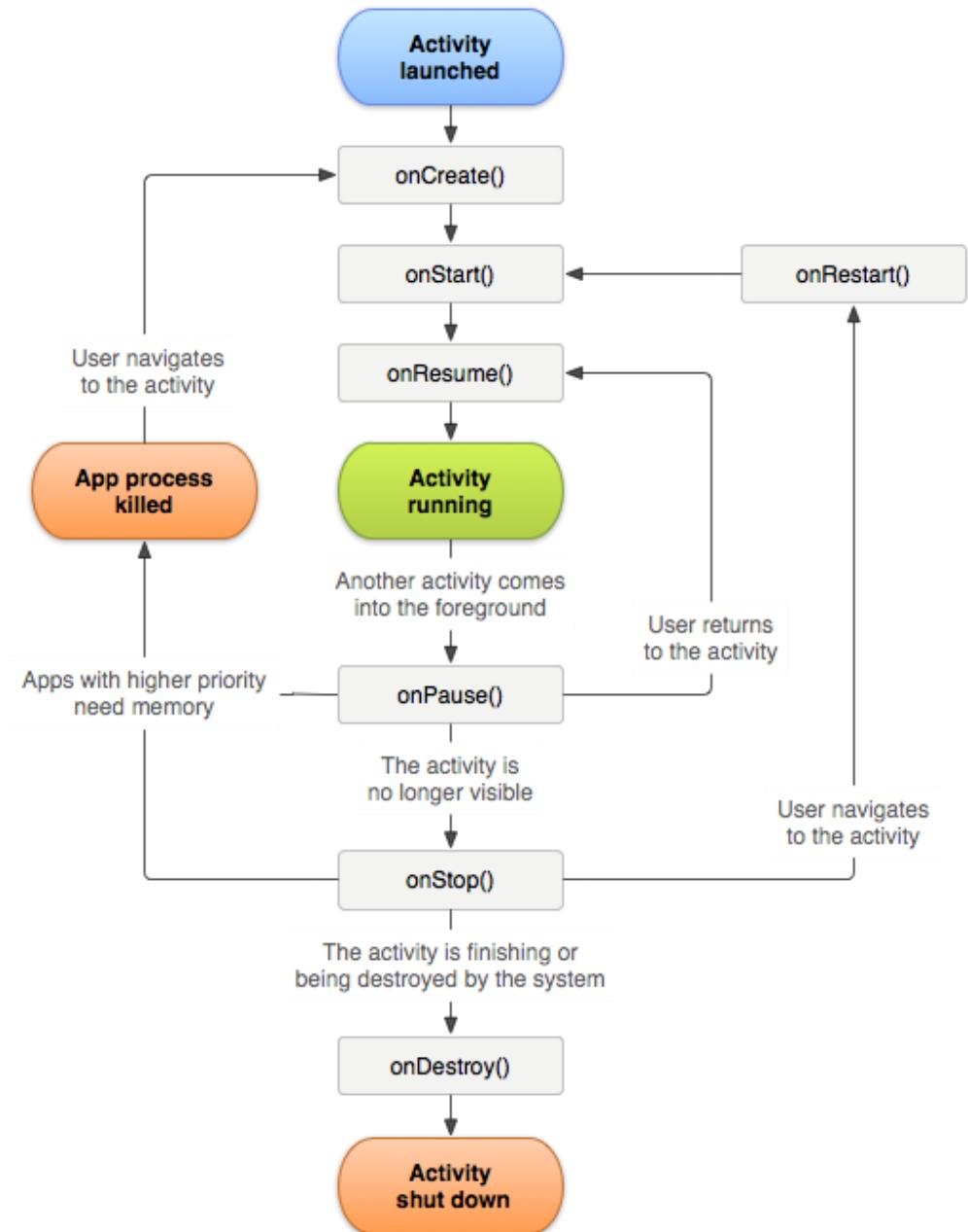
Storing Activity State in Activity Records

- Android provides a way to save and retrieve activity state at time of transitions:
 - `onSaveInstanceState (Bundle)` saves current instance state in an **activity record**
 - `onCreate (Bundle)` reconstitutes the activity's state from the saved activity record
- An activity record is kept even if the activity instance is removed from memory
 - Can be used to recreate an activity in its previous state



Caution: Simplified Introduction to Activity Lifecycle

- The preceding introduction has been simplified for the sake of clarity.
- For a precise specification of the circumstances under which the various lifecycle and state-saving methods are called, see
 - <http://developer.android.com/training/basics/activity-lifecycle/startling.html>
 - <http://developer.android.com/reference/android/app/Activity.html>



Storing Activity State in onSaveInstanceState

- Method called by Android when activity becomes killable
 - i.e. when it reaches a state where it may be removed from memory without further notice
 - Default implementation lets all view objects store their state in a **Bundle**
- We can extend the method to store additional data in the same **Bundle**
- A **Bundle** is a data structure mapping keys to values
 - Keys are string constants
 - Values are typically primitive types
 - Serializable** or **Parcelable** classes are possible but discouraged

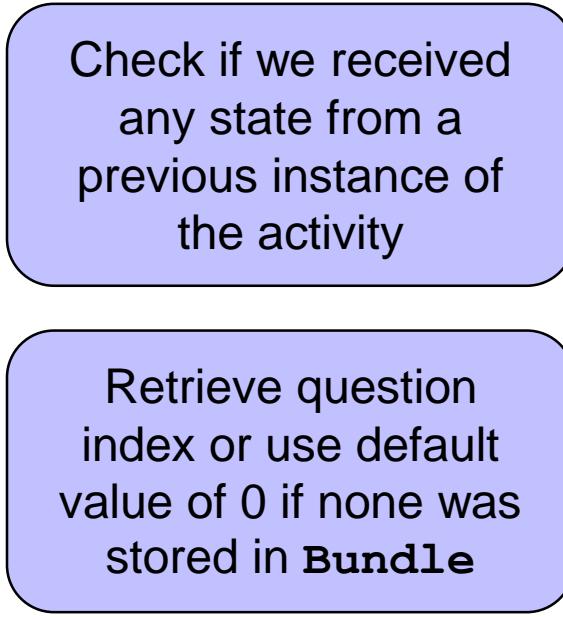
```
public class MainActivity extends AppCompatActivity {  
  
    private static final String KEY_INDEX = "index";  
  
    private int mCurrentIndex = 0;  
  
    // ...  
    @Override  
    public void onSaveInstanceState(Bundle savedInstanceState) {  
        super.onSaveInstanceState(savedInstanceState);  
        savedInstanceState.putInt(KEY_INDEX, mCurrentIndex);  
    }  
    // ...  
}
```



Recovering Activity State in onCreate

- Android provides a **Bundle** containing previously stored activity state (if any)
- We can recover previously stored values from this (if present)

```
public class MainActivity extends AppCompatActivity {  
  
    private static final String KEY_INDEX = "index";  
    private int mCurrentIndex = 0;  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        // ...  
        if (savedInstanceState != null) {  
            mCurrentIndex = savedInstanceState.getInt(KEY_INDEX, 0);  
        }  
        // ...  
    }  
    // ...  
}
```



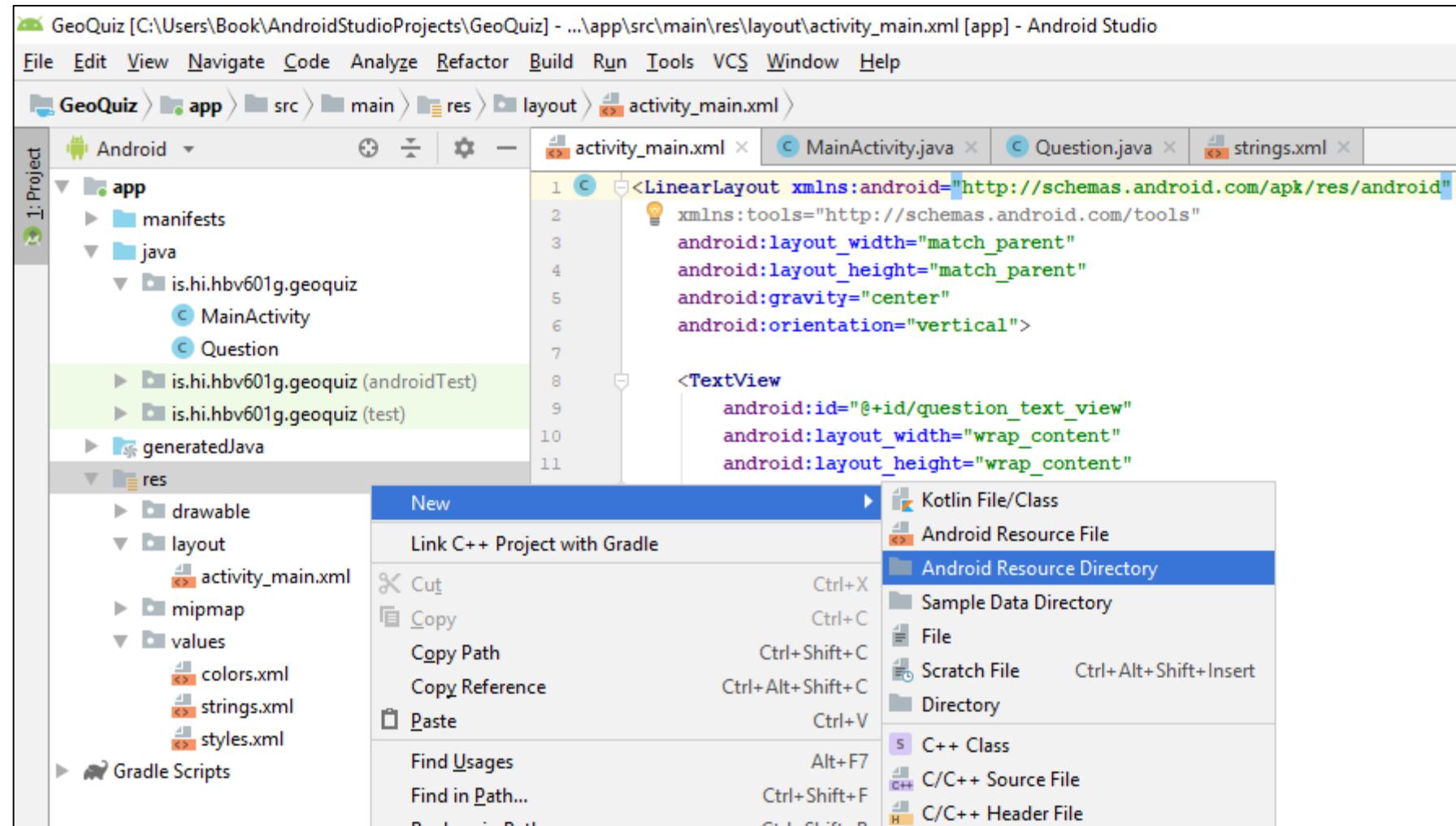
What to Do Where?

- **onCreate** is typically overridden to prepare the activity's user interface:
 - Inflating widgets and putting them on screen (with `setContentView`)
 - Getting references to inflated widgets, in order to work with them later
 - Setting listeners on widgets to handle user interaction
 - Retrieving saved instance state
 - Connecting to external model data
- **onSaveInstanceState** is typically overridden to store small, transient items
- **onPause** is typically overridden for larger save/cleanup tasks upon loss of focus
- What you do in **onPause**, **onStop**, **onDestroy** etc. depends on your activity:
 - How will people probably use it?
 - How will people probably leave it?
 - How might people get interrupted?
 - How might people resume it?
 - How would people expect to find it upon returning?



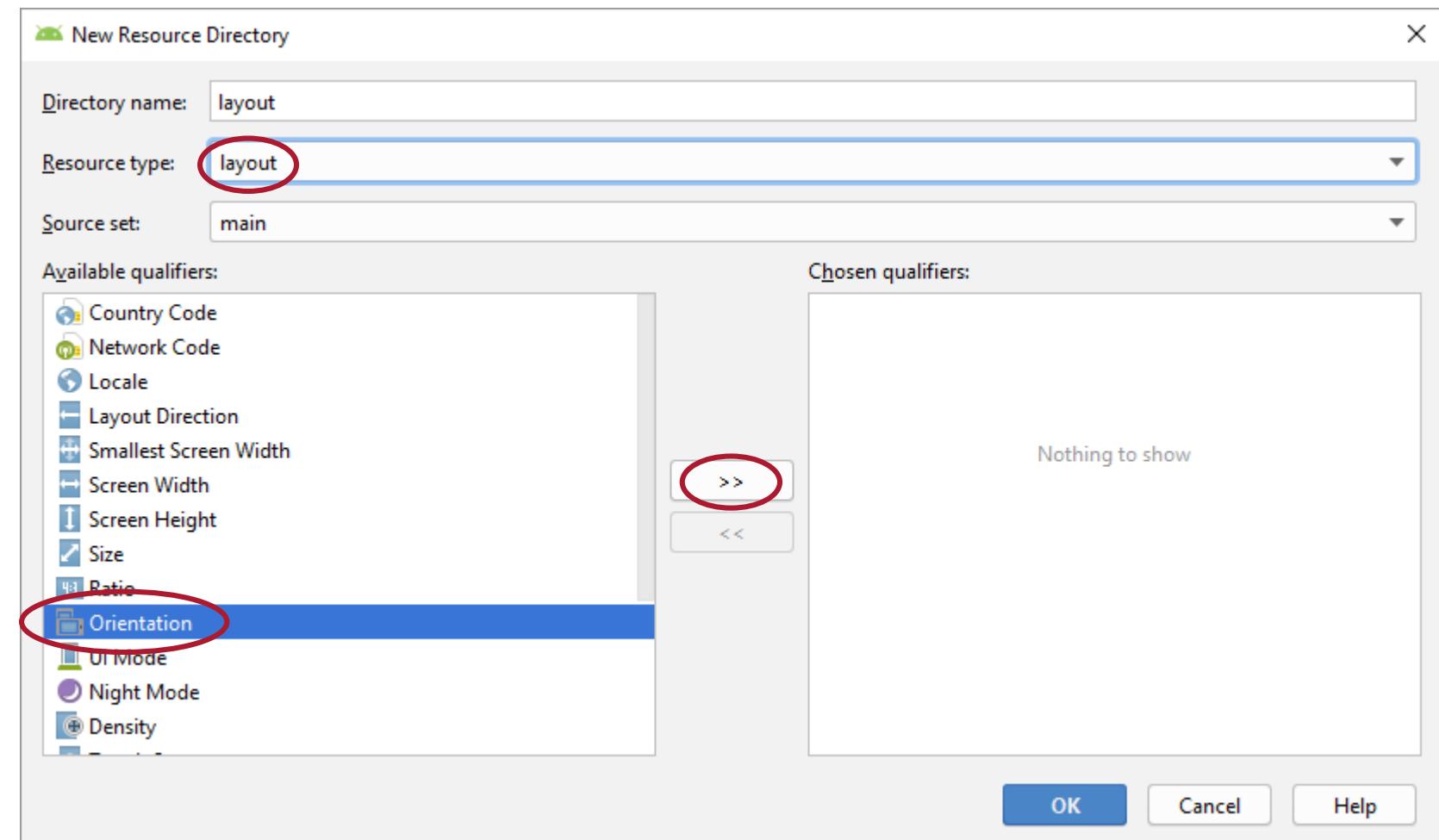
Adding a Different Layout for Landscape Orientation

- Portrait layout used so far was stored in **res/layout**
- Android expects landscape layout in **res/layout-land**
- To create, right-click **res** folder; choose “New > Android Resource Directory”



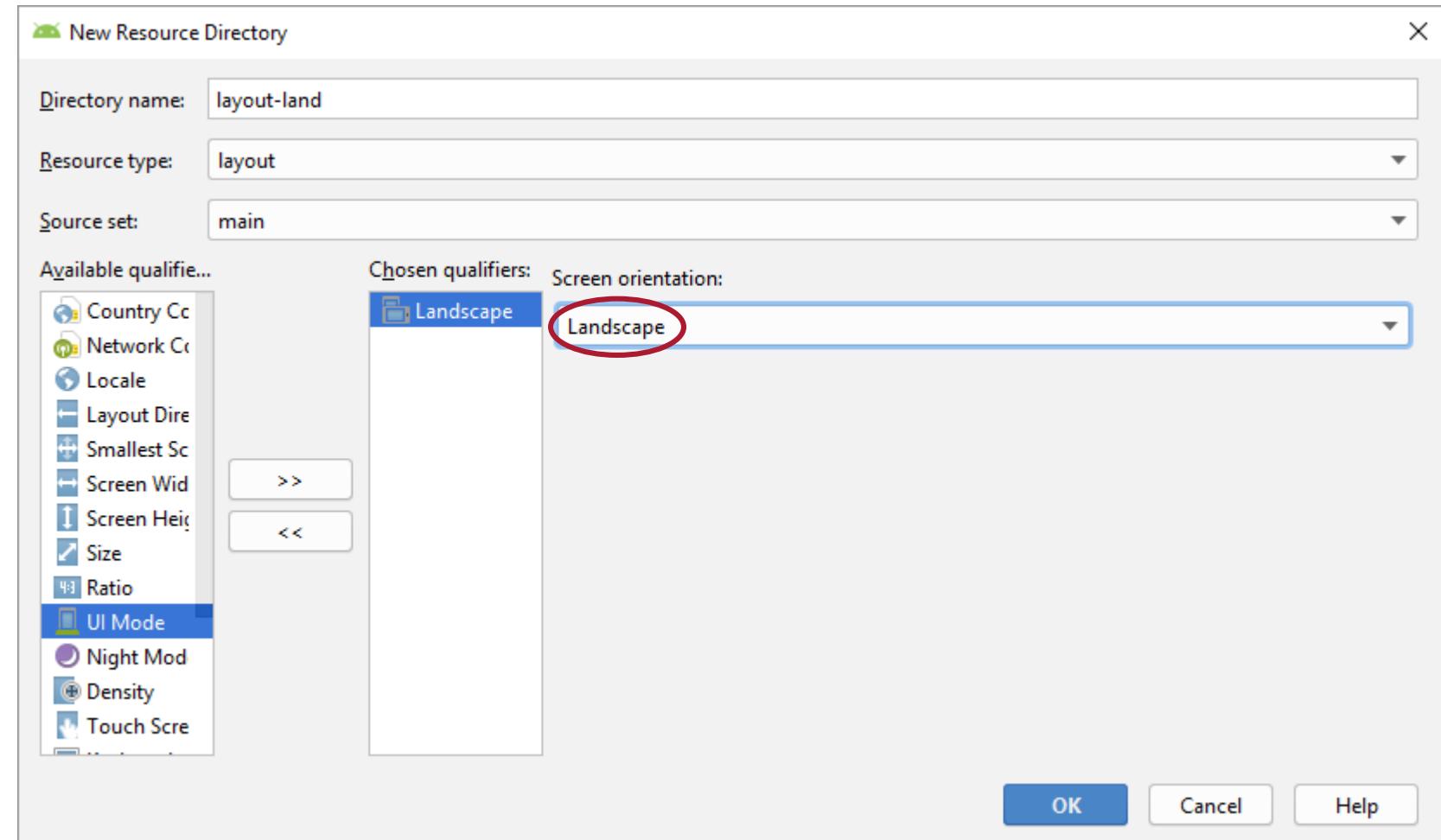
Adding a Different Layout for Landscape Orientation

- Choose resource type “layout”
 - Note directory name is updated automatically
- Select qualifier “Orientation” and click “>>” button



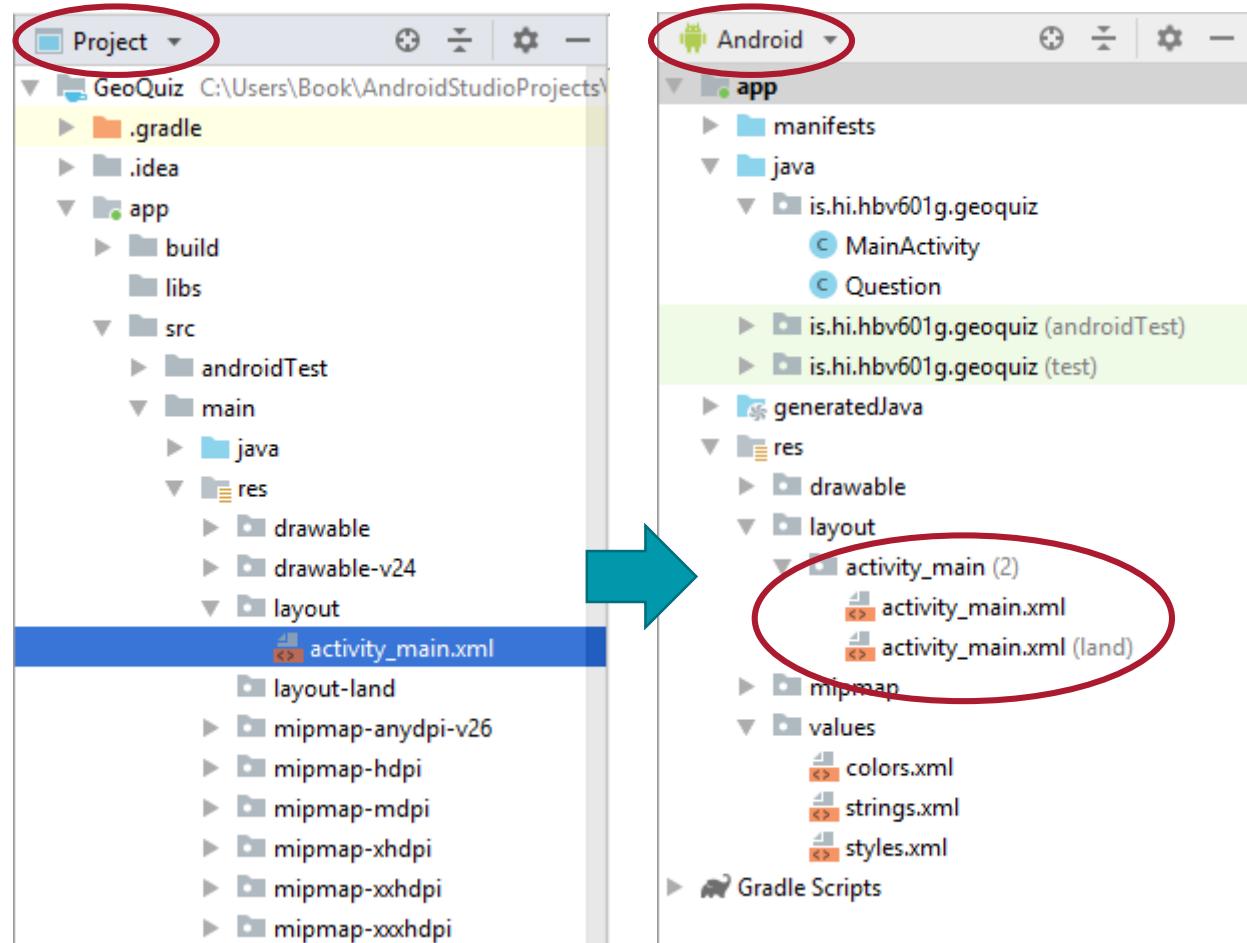
Adding a Different Layout for Landscape Orientation

- Select screen orientation “Landscape”
 - Note directory name is updated accordingly again
- Click OK button
- This will create the additional folder **res/layout-land** beside the original **layout** resource folder



Configuration Qualifiers

- `land` suffix is one of many **configuration qualifiers** helping Android to find resources that best match a given device configuration
 - Many more folder types and qualifiers possible, e.g. different image resolutions for different screen densities
 - <http://developer.android.com/guide/topics/resources/providing-resources.html>
- Note: New folder doesn't show up in IDE's *Android* perspective while empty
 - Use *Project* perspective to copy layout file `activity_main.xml` from `layout` folder to `layout-land` folder



Defining a Landscape Layout (Layout-land/activity_main.xml)

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">

    <TextView
        android:id="@+id/question_text_view"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:padding="24dp"/>

    <LinearLayout
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_gravity="center_vertical|center_horizontal"
        android:orientation="horizontal">

        <Button android:id="@+id/true_button" .../>
        <Button android:id="@+id/false_button" .../>
    </LinearLayout>

    <Button
        android:id="@+id/next_button"
        android:layout_width="wrap_content" android:layout_height="wrap_content"
        android:layout_gravity="bottom|right"
        android:text="@string/next_button"/>

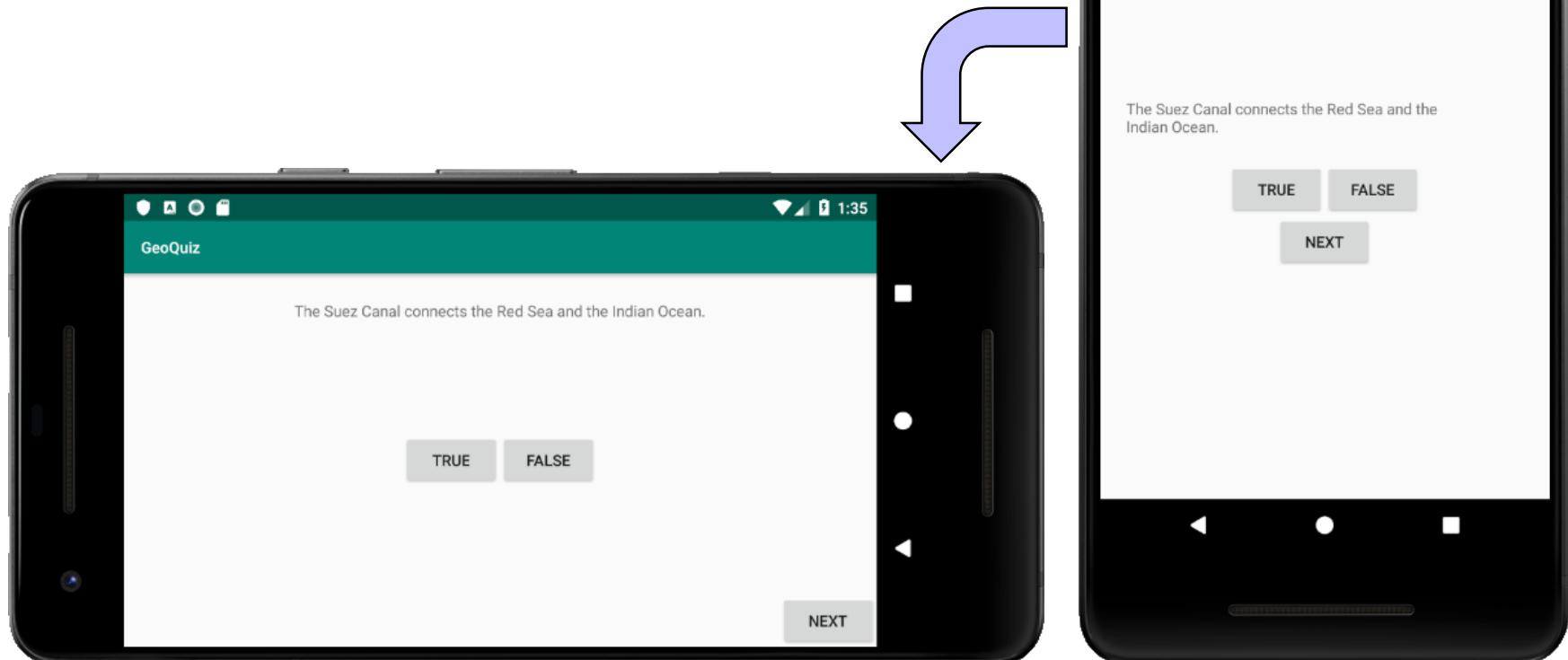
</FrameLayout>
```

Indicates where this view should be placed in enclosing view



Testing the Revised Activity and New Layout

- When rotating the device...
 - ✓ we see the new landscape layout
 - ✓ the newly created MainActivity uses the previously saved question index

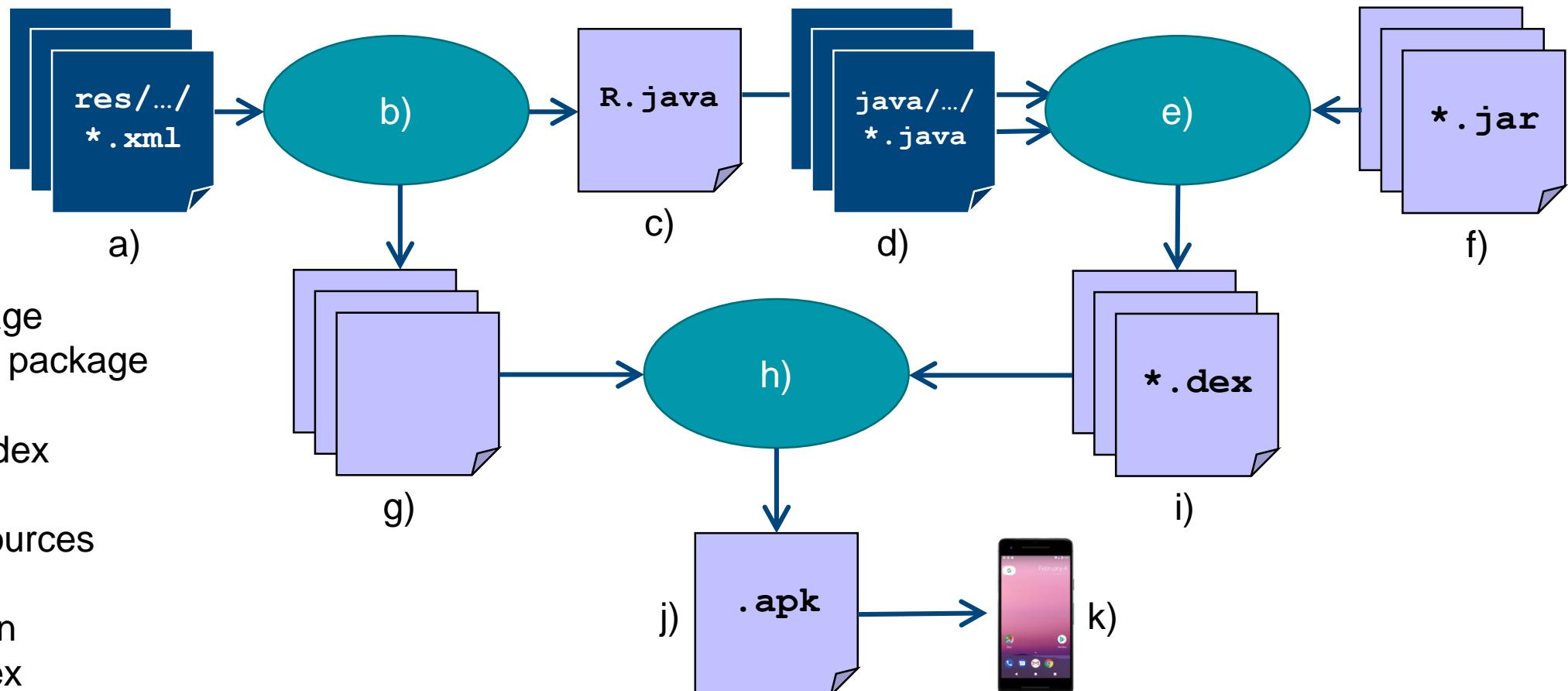


In-Class Quiz #4: The Android Build Process



Assign the proper labels to the elements of the build process:

1. Android package
2. Build and sign package
3. Bytecode
4. Collect and index
5. Compile
6. Compiled resources
7. Dependencies
8. Deploy and run
9. Resource index
10. Resources
11. Source code





Hugbúnaðarverkefni 2 / Software Project 2

7. Android User Interfaces

HBV601G – Spring 2020

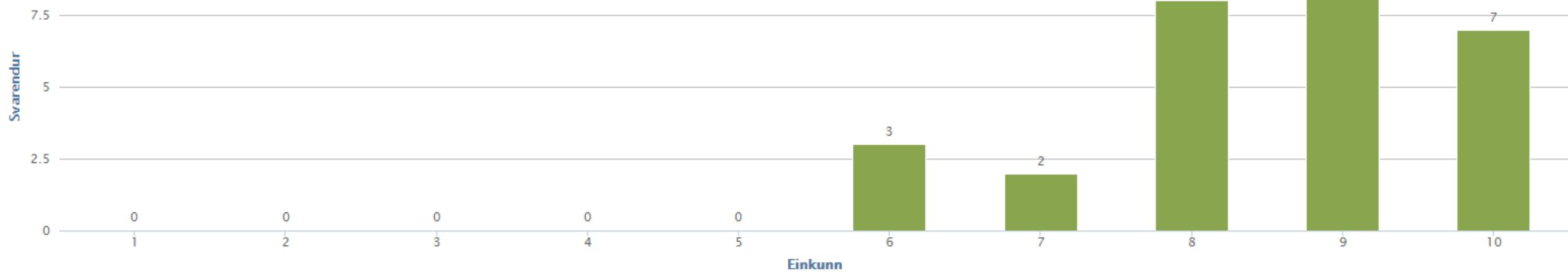
Matthias Book



HÁSKÓLI ÍSLANDS
VERKFRAÐI- OG NÁTTÚRVÍSINDASVIÐ
ÍÐNAÐARVERKFRAÐI-, VÉLAVERKFRAÐI-
OG TÖLVUNARFRÆÐIDEILD

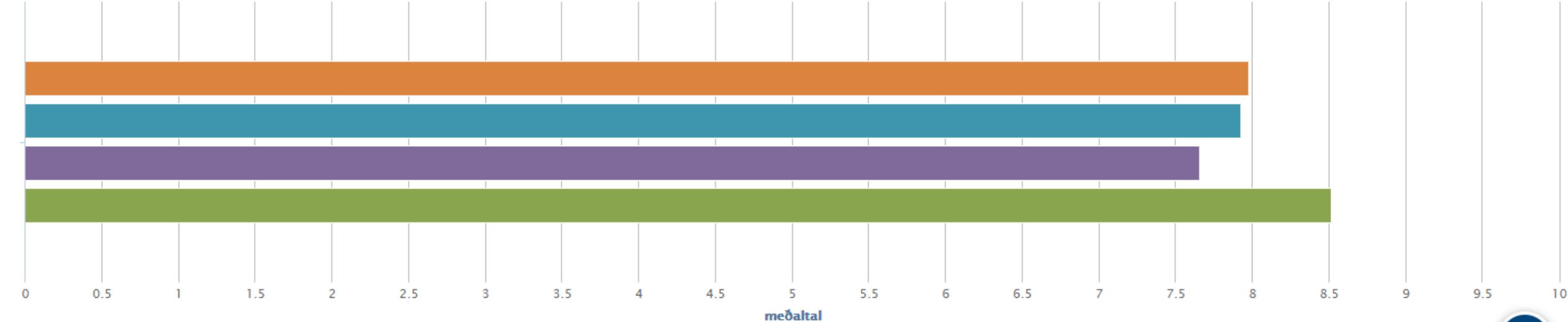
Mid-Semester Evaluation

Gefðu námskeiðinu einkunn:
Dreifing einkunna



Takk fyrir!

Gefðu námskeiðinu einkunn:
Samanburður



Participation: 29/87

HBV601G Iðnaðarverkfræði-, vélaverkfræði- og tölvunarfræðideild Verkfræði- og náttúruvísindasvið Háskóli Íslands



Feedback (paraphrased)

Likes

- Well-organized course
- Scheduling flexibility
- Good teaching

- HBV401-601G courses feel repetitive
 - Courses address same topics from different angles / at different depths
 - Stretching coverage of all project phases across three semesters would make project work impossible
 - A lot of engineering skill is experience

Dislikes

- Frustration about inactive team members
 - Escalate to your tutor and/or me as early as possible!

- Having to use plain Java when there are UI frameworks available
 - Frameworks and languages come and go
 - Important to be familiar with the fundamentals going on behind the scenes, in order to make good technology decisions in later projects



In-Class Quiz 5 Prep

- Please prepare a small scrap of paper with the following information:

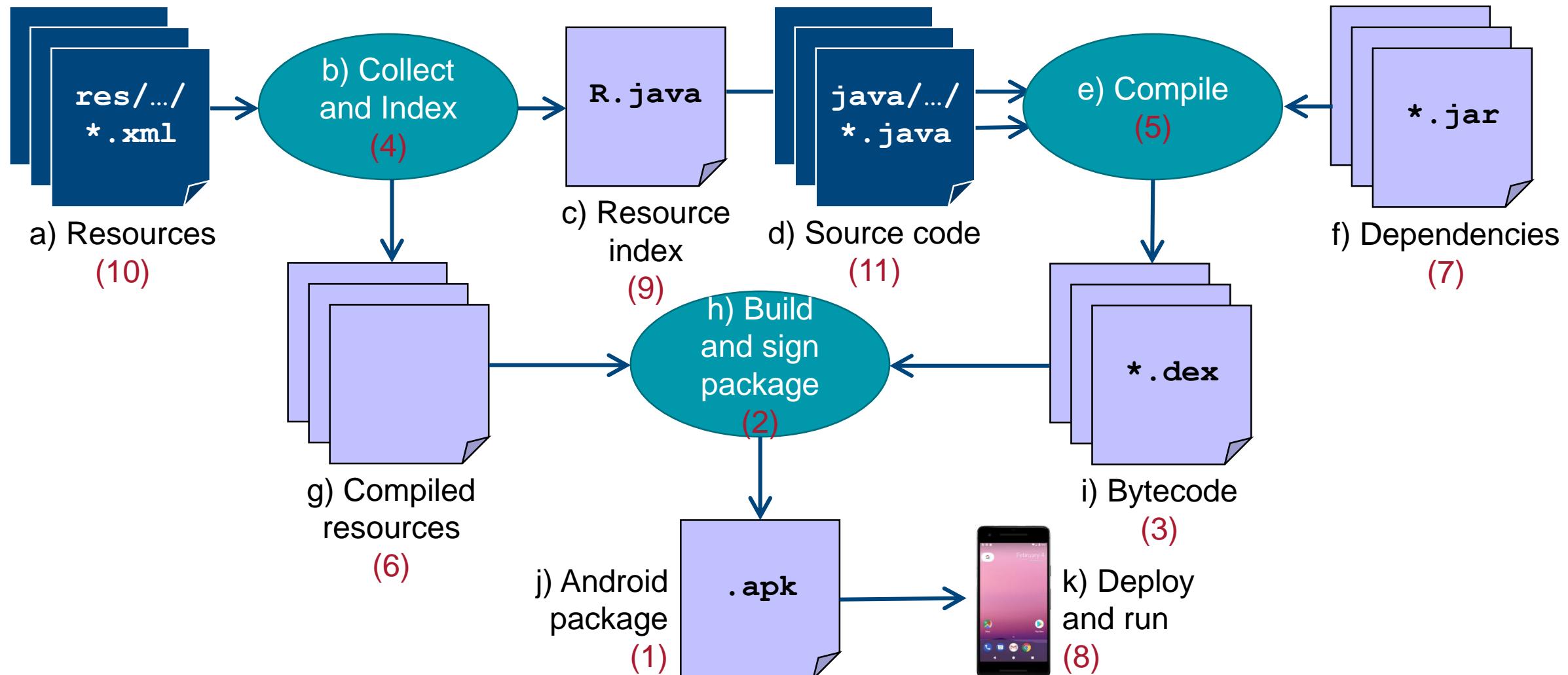
ID: _____ @hi.is Date: _____

a) _____ d) _____
b) _____ e) _____
c) _____ f) _____

- During class, I'll show you questions that you can answer with a number
- Hand in your scrap at end of class
- All questions in a quiz have same weight
- All quizzes (8-10 throughout semester) have the same weight
 - Your worst 2 quizzes will be disregarded
- Overall quiz grade counts as optional question worth 7.5% on final exam

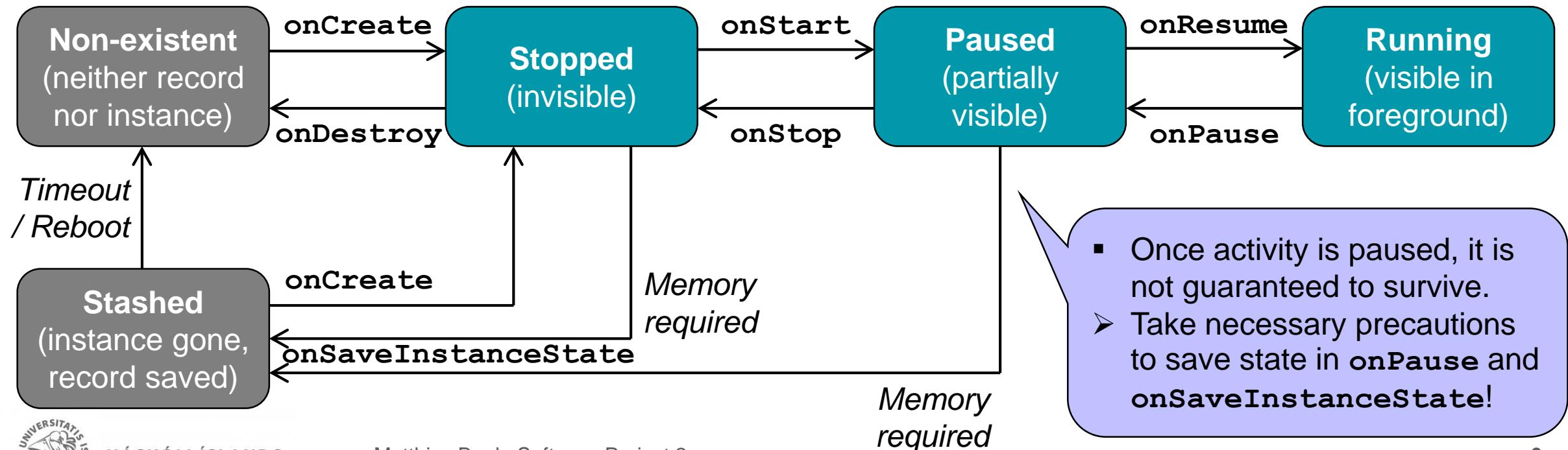


In-Class Quiz #4 Solution



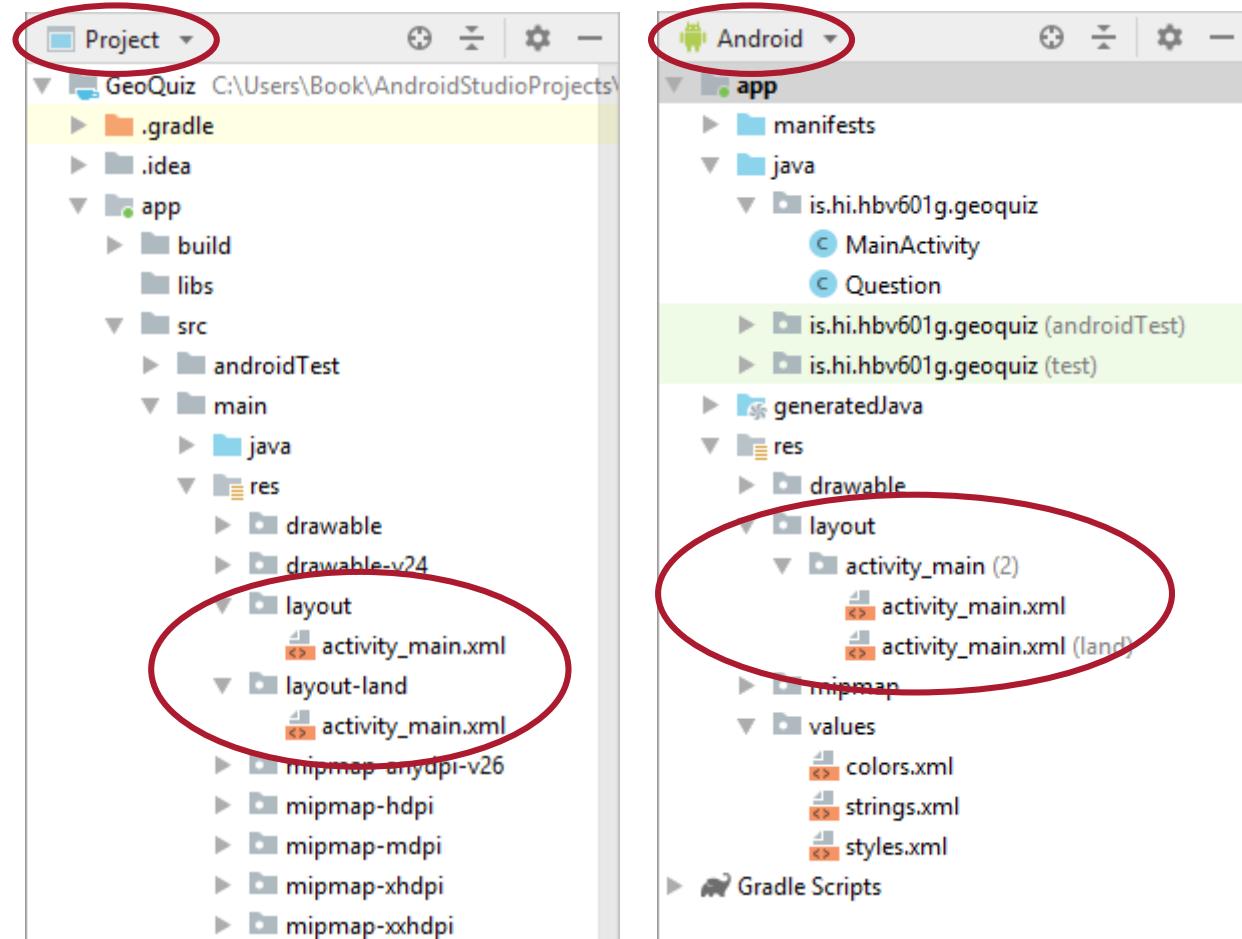
Recap: Storing Activity State in Activity Records

- Android provides a way to save and retrieve activity state at time of transitions:
 - `onSaveInstanceState (Bundle)` saves current instance state in an **activity record**
 - `onCreate (Bundle)` reconstitutes the activity's state from the saved activity record
- An activity record is kept even if the activity instance is removed from memory
 - Can be used to recreate an activity in its previous state



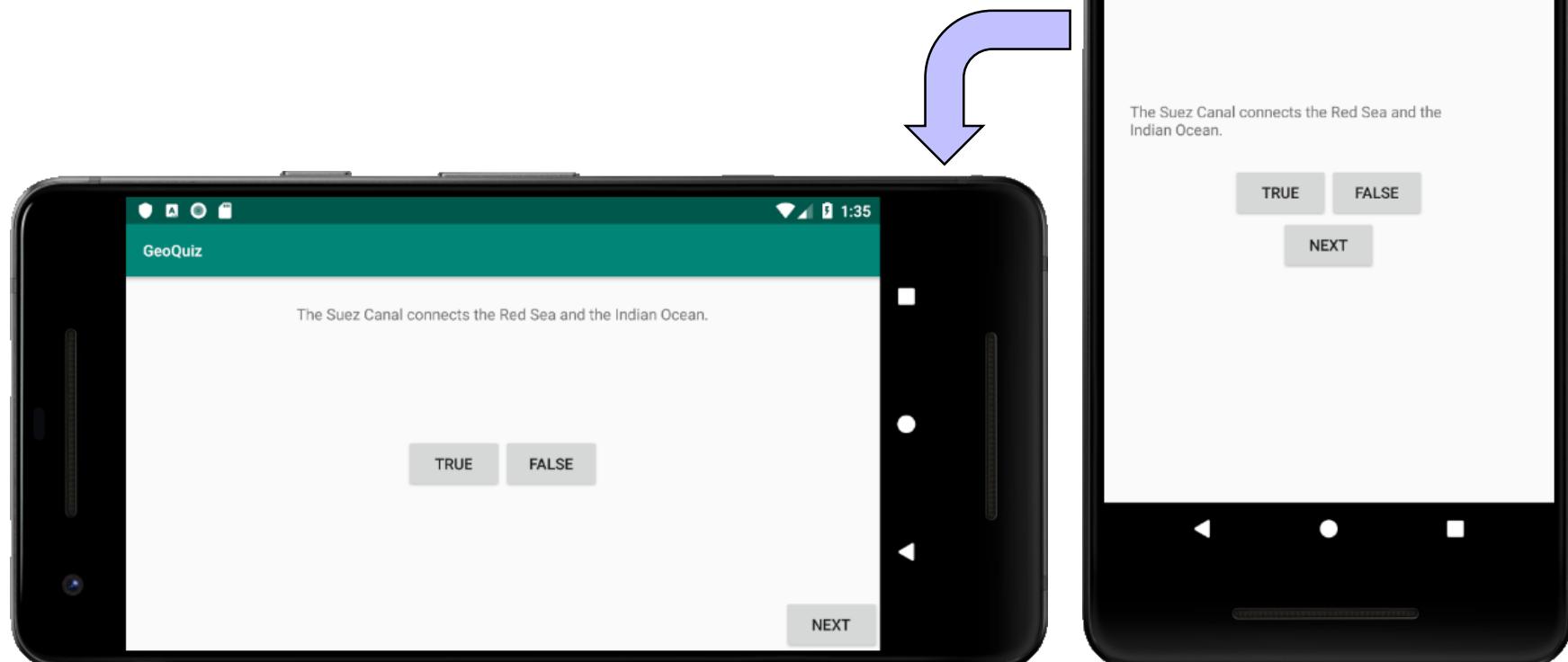
Recap: Configuration Qualifiers

- Portrait layout used so far was stored in **res/layout**
- Android expects landscape layout in **res/layout-land**
- **-land suffix is a configuration qualifier** helping Android to find resources that best match a given device configuration
 - Many more folder types and qualifiers possible, e.g. different image resolutions for different screen densities
 - <http://developer.android.com/guide/topics/resources/providing-resources.html>



Recap: GeoQuiz App

- When rotating the device...
 - ✓ we see the new landscape layout
 - ✓ the newly created MainActivity uses the previously saved question index



Communicating With Intents

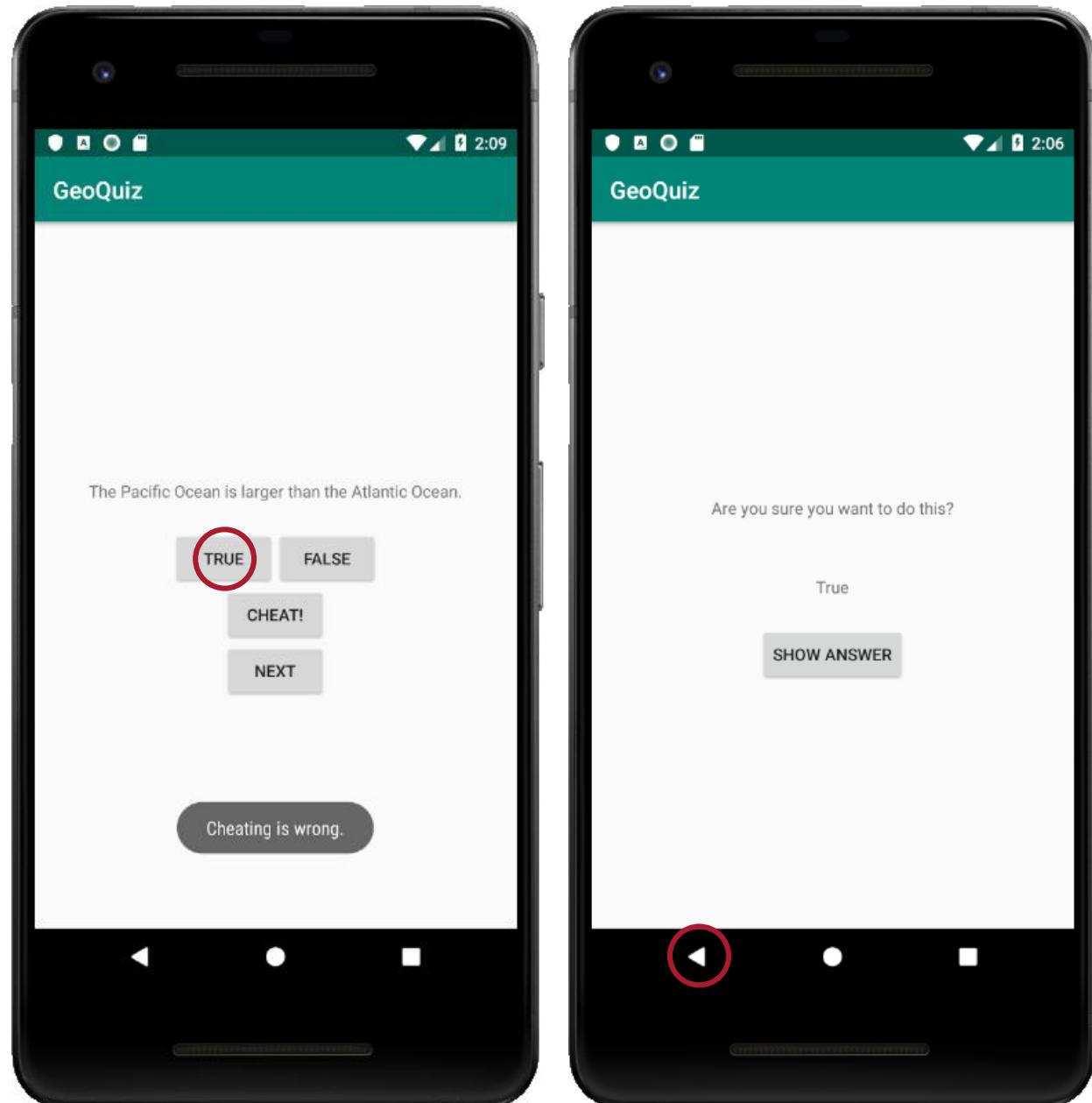
see also:

- Phillips et al.: Android Development, Ch. 5
- <http://developer.android.com/training/basics/firstapp/starting-activity.html>
- <http://developer.android.com/guide/components/intents-filters.html>



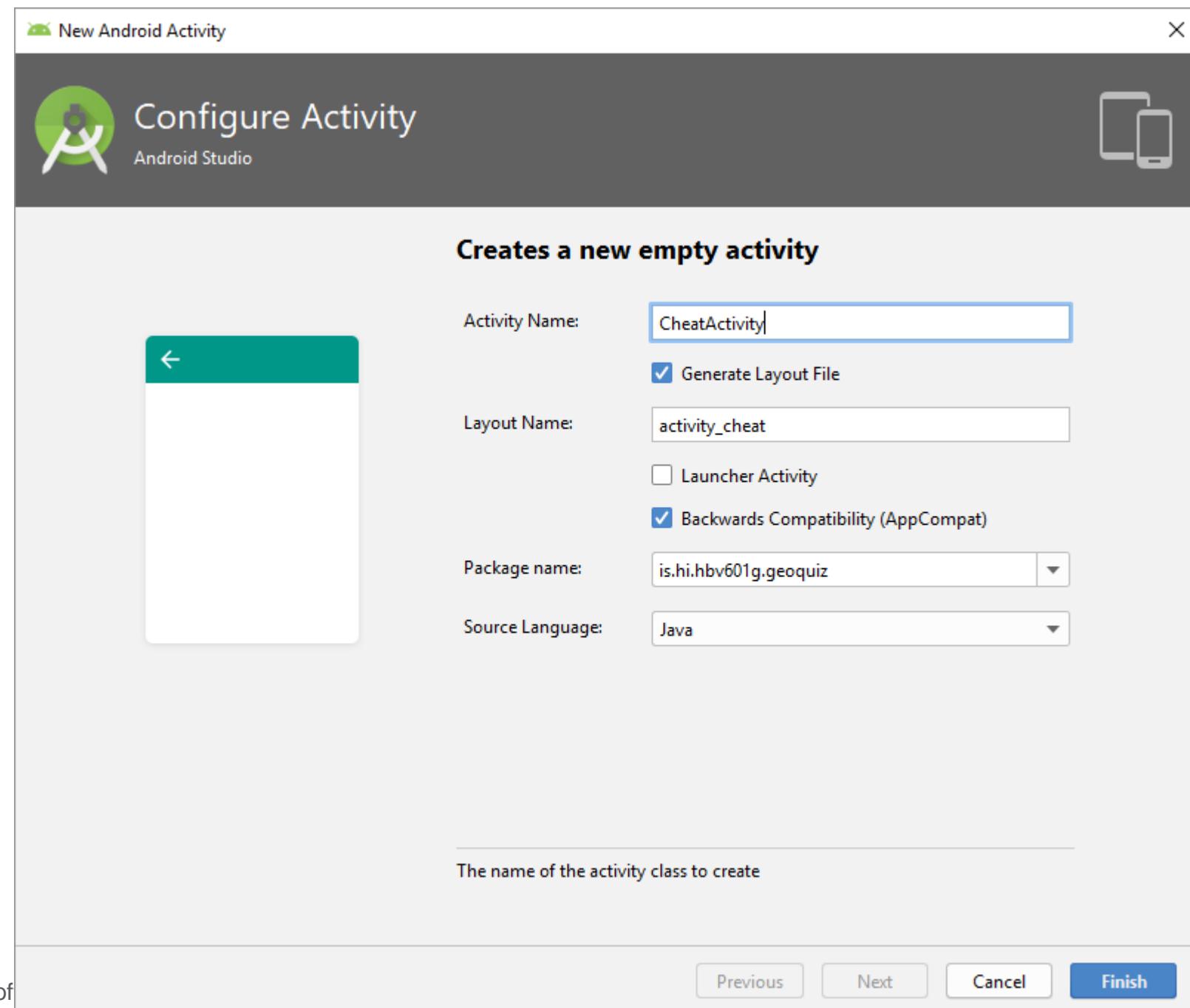
Example Scenario

- On each question screen, we want to give the user the option to “cheat” by looking up the right answer on another screen, before returning to the question screen.
- That means we need to
 - invoke another screen (i.e. activity)
 - and pass information to it;
 - then return to the previous activity
 - and return information to it.
- This is accomplished with **intents**.



Creating a New Activity

- Choose “File > New > Activity > Empty Activity”
- Provide a name for the activity
 - here: CheatActivity for the activity governing the cheat screen’s behavior
- Layout name will be set automatically, following naming convention
 - ...and layout file will be created together with activity file



Explicit Intents

- An activity cannot directly invoke another activity by calling its methods.
- Rather, an activity communicates with the Android Operating System (OS) by exchanging **intents**.
- To invoke another activity within the same app, we use an **explicit intent**:

Context we are calling from

Activity to invoke

```
Intent i = new Intent(MainActivity.this, CheatActivity.class);
i.putExtra("is.hi.hbv601g.geoquiz.answer_is_true", answerIsTrue);
startActivity(i);
```

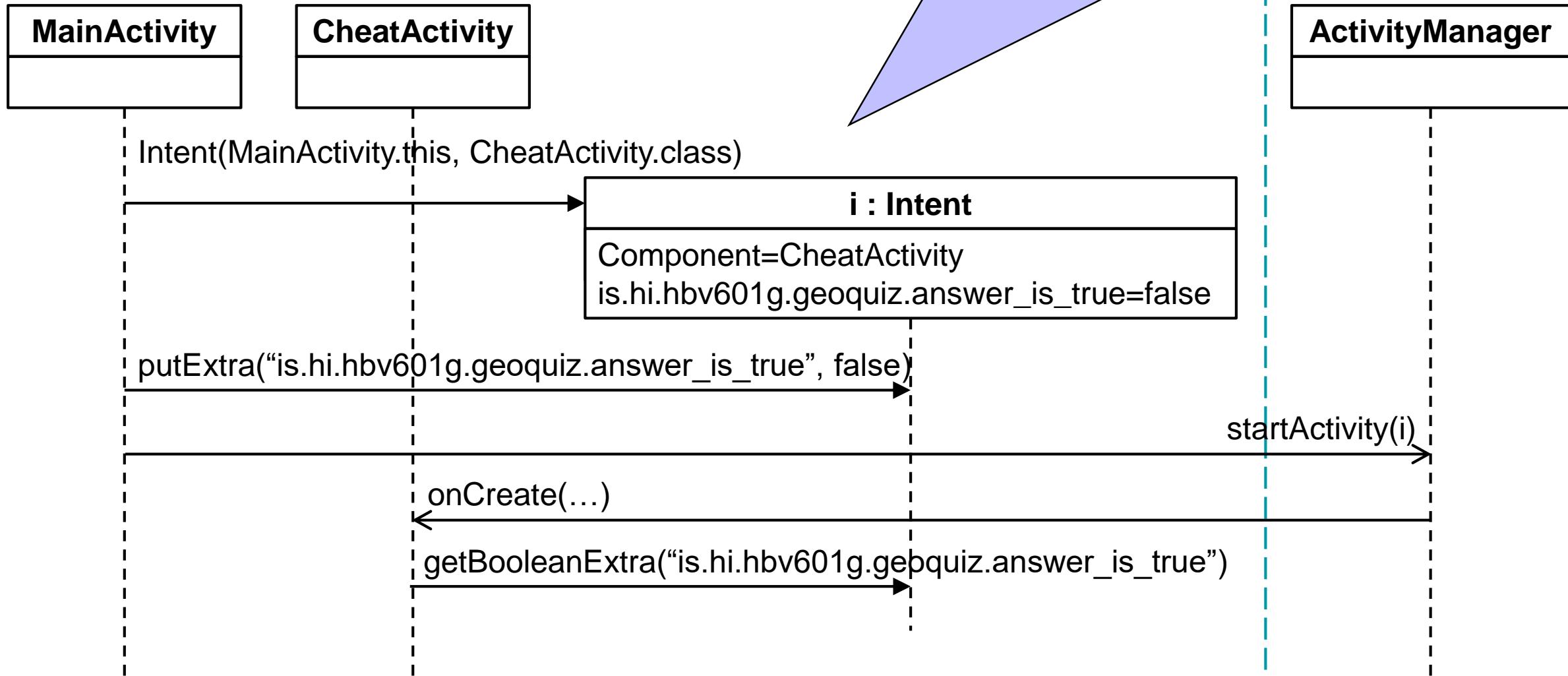
Extra's key

Extra's value

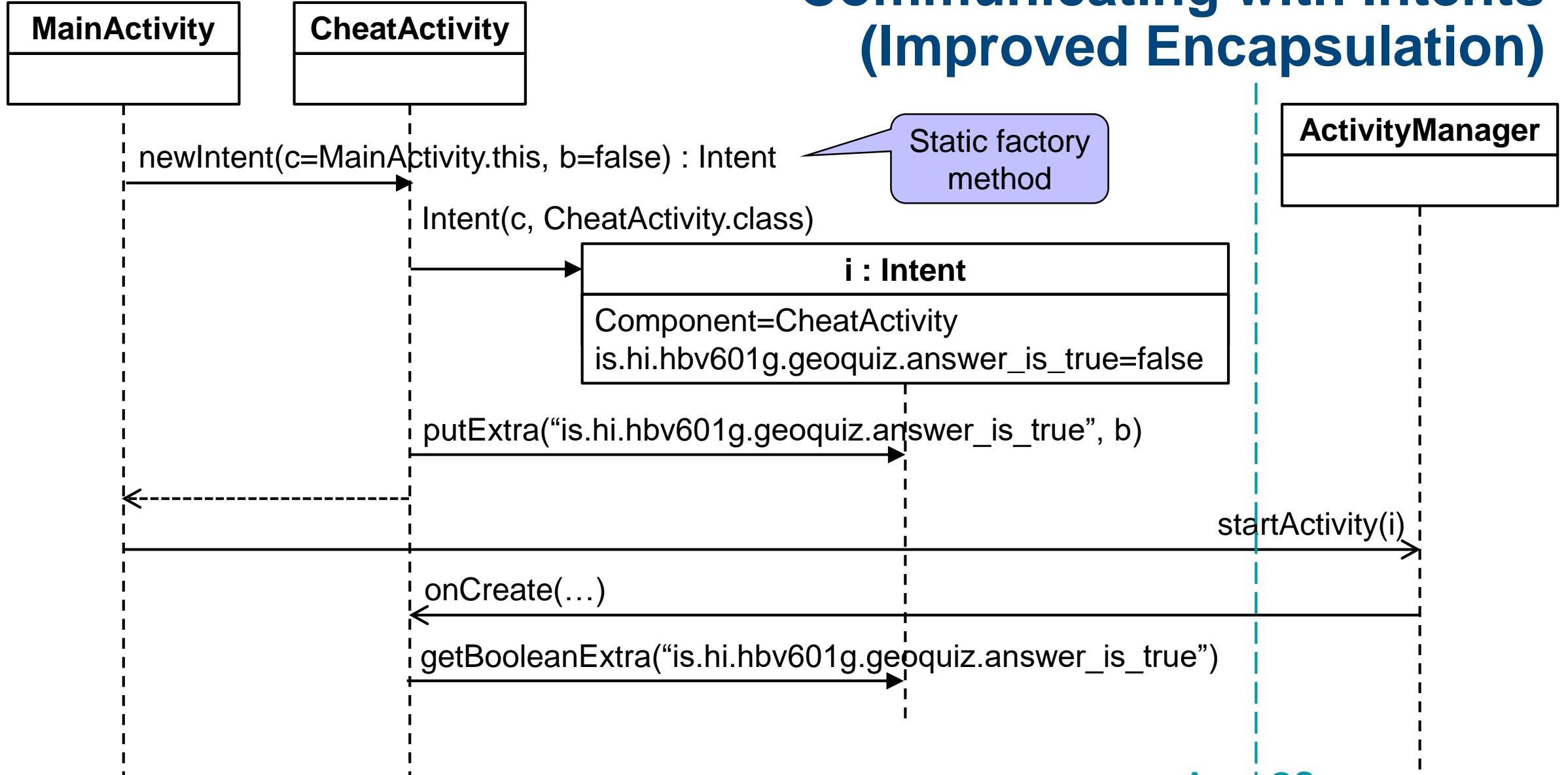
- **Extras** are arbitrary data (key-value pairs) included with the intent.
- We can also invoke activities in other apps using **implicit intents** (*covered later*)

Communicating with Intents

Inelegant: The extra's keys and types constitute an interface of `CheatActivity`! `MainActivity` should not have to be aware of its implementation details.



Communicating with Intents (Improved Encapsulation)



Defining an Intent for Other Activities to Use (CheatActivity.java)

```
public class CheatActivity extends AppCompatActivity {  
  
    private static final String EXTRA_ANSWER_IS_TRUE =  
        "is.hi.hbv601g.geoquiz.answer_is_true";  
  
    public static Intent newIntent(Context packageContext, boolean answerIsTrue) {  
        Intent i = new Intent(packageContext, CheatActivity.class);  
        i.putExtra(EXTRA_ANSWER_IS_TRUE, answerIsTrue);  
        return i;  
    }  
  
    // ...  
}
```

Use a constant instead of literal to ensure we use the same key when putting and getting the extra

- CheatActivity provides static method `newIntent` that determines what the intents that it expects will look like:
 - a `Context` parameter (the calling activity)
 - a `boolean` parameter (the correct answer)
- No need for other activities to construct intents and extras, or to know what the extra should be called

Invoking a Child Activity in Order to Get a Result Back

- To get a result back, don't invoke an activity with `startActivity(Intent i)` but with `startActivityForResult(Intent i, int requestCode)`
- **requestCode** is an arbitrary integer that your activity will receive back with the result, in order to recognize which request the result refers to
 - since your activity may invoke multiple other activities, and doesn't know when it'll hear back



Using the Intent to Invoke CheatActivity (MainActivity.java)

```
public class MainActivity extends AppCompatActivity {

    private static final int REQUEST_CODE_CHEAT = 0;

    // ...

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        // ...
        mCheatButton = (Button) findViewById(R.id.cheat_button);
        mCheatButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                boolean answerIsTrue = mQuestionBank[mcurrentIndex].isAnswerTrue();
                Intent i = CheatActivity.newIntent(MainActivity.this, answerIsTrue);
                startActivityForResult(i, REQUEST_CODE_CHEAT);
            }
        });
        // ...
    }
}
```



Accessing the Extras in the Received Intent (CheatActivity.java)

```
public class CheatActivity extends AppCompatActivity {

    private static final String EXTRA_ANSWER_IS_TRUE =
        "is.hi.hbv601g.geoquiz.answer_is_true";

    private boolean mAnswerIsTrue;

    // ...

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

        mAnswerIsTrue = getIntent().getBooleanExtra(EXTRA_ANSWER_IS_TRUE, false);
    }

    // ...
}
```

Retrieve the intent that started this activity

Retrieve the extra expected in the intent

Default value if extra is not defined in intent



Sending a Result Back From a Child Activity in an Intent

- [Not shown before for brevity: Code for revealing the answer]
- To return data to the parent activity, perform these steps in the child activity:
 - Create another intent
 - Put the data you want to return into it as an extra
 - Call the `setResult` method in order to create the intent that will be sent back to the caller:
`public final void setResult(int resultCode, Intent data)`
 - `resultCode`: can be set to `Activity.RESULT_OK` or `Activity.RESULT_CANCELED`
 - `data`: the intent containing your custom result data (optional)
 - The intent will be sent by Android when the user leaves the child activity (i.e. backs out of it)
- Example scenario:
 - We want to inform `MainActivity` about whether the user actually peeked at the answer in `CheatActivity`, so we can react accordingly.

Returning a Result in an Intent (and Accessing it) (CheatActivity.java)

```
public class CheatActivity extends AppCompatActivity {  
  
    private static final String EXTRA_ANSWER_SHOWN =  
        "is.hi.hbv601g.geoquiz.answer_shown";  
  
    private void setAnswerShownResult(boolean isAnswerShown) {  
        Intent data = new Intent();  
        data.putExtra(EXTRA_ANSWER_SHOWN, isAnswerShown);  
        setResult.RESULT_OK, data);  
    }  
  
    public static boolean wasAnswerShown(Intent result) {  
        return result.getBooleanExtra(EXTRA_ANSWER_SHOWN, false);  
    }  
    // ...  
}
```

Called by the event handler for clicks on the “Show Answer” button

Put an extra in the return intent to indicate whether the answer has been revealed

Static method to help parent activity access the content of the intent’s extras without having to know about their label and type



Receiving the Result Data in the Parent Activity

- Android will call the `onActivityResult` event handler on the parent activity in order to send the result back to it:

```
protected void onActivityResult(int requestCode,  
                               int resultCode,  
                               Intent data)
```

- `requestCode` is the code you set in your original intent invoking the child activity
 - `resultCode` is the code set by the child activity
 - `data` is the intent containing the child activity's custom result data
-
- Since the structure of the result intent is another interface of the child activity, that activity should provide a method for interpreting the intent's extras
 - So the parent activity doesn't need to take the intent apart and be dependent on [= exposed to changes of] the extras' implementation details

Receiving and Reacting to the Result (MainActivity.java)

```
public class MainActivity extends AppCompatActivity {  
  
    private boolean mIsCheater;  
  
    @Override  
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
        if (resultCode != Activity.RESULT_OK) {  
            return;  
        }  
        if (requestCode == REQUEST_CODE_CHEAT) {  
            if (data == null) {  
                return;  
            }  
            mIsCheater = CheatActivity.wasAnswerShown(data);  
        }  
    }  
}  
// ...
```

Called by Android when an intent with a result comes in

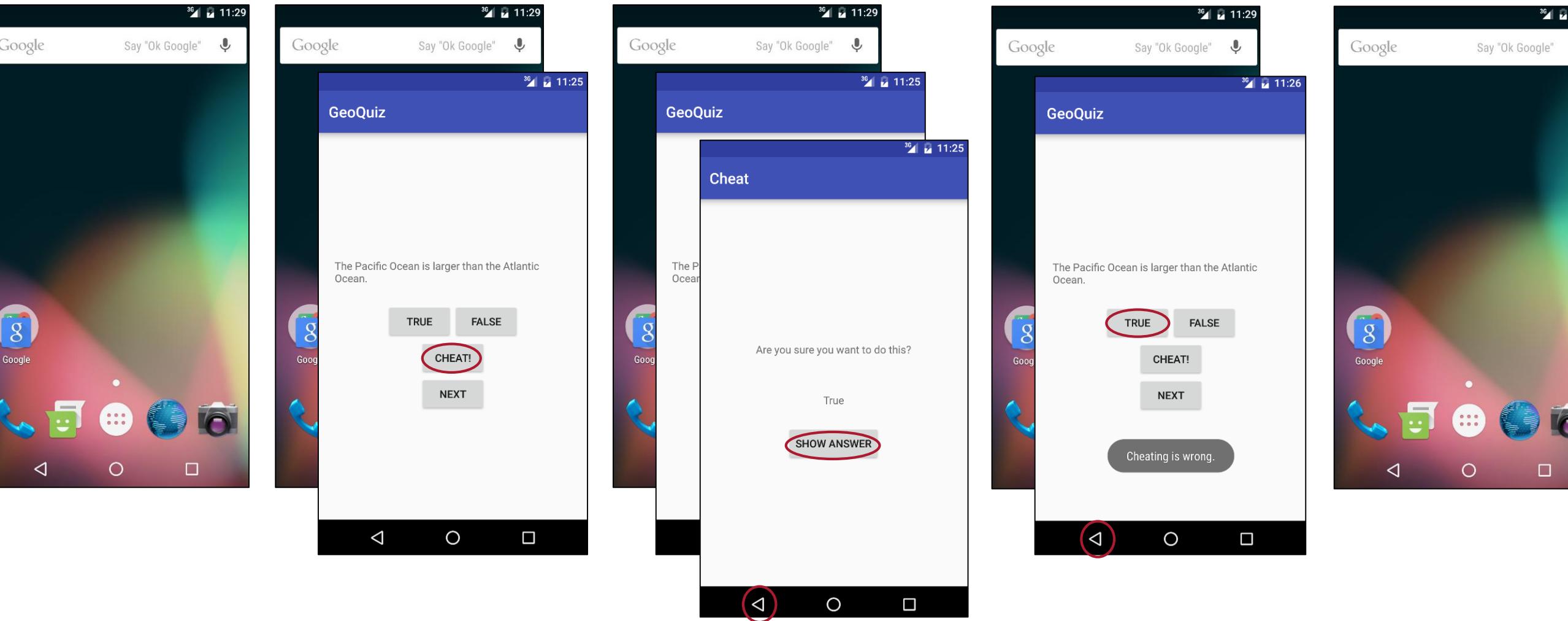
Check whether this event has the request and result codes we're expecting

Retrieve result data using the "interpreter" method provided by the child activity

[Not shown for brevity: Code for calling out the user as a cheater]



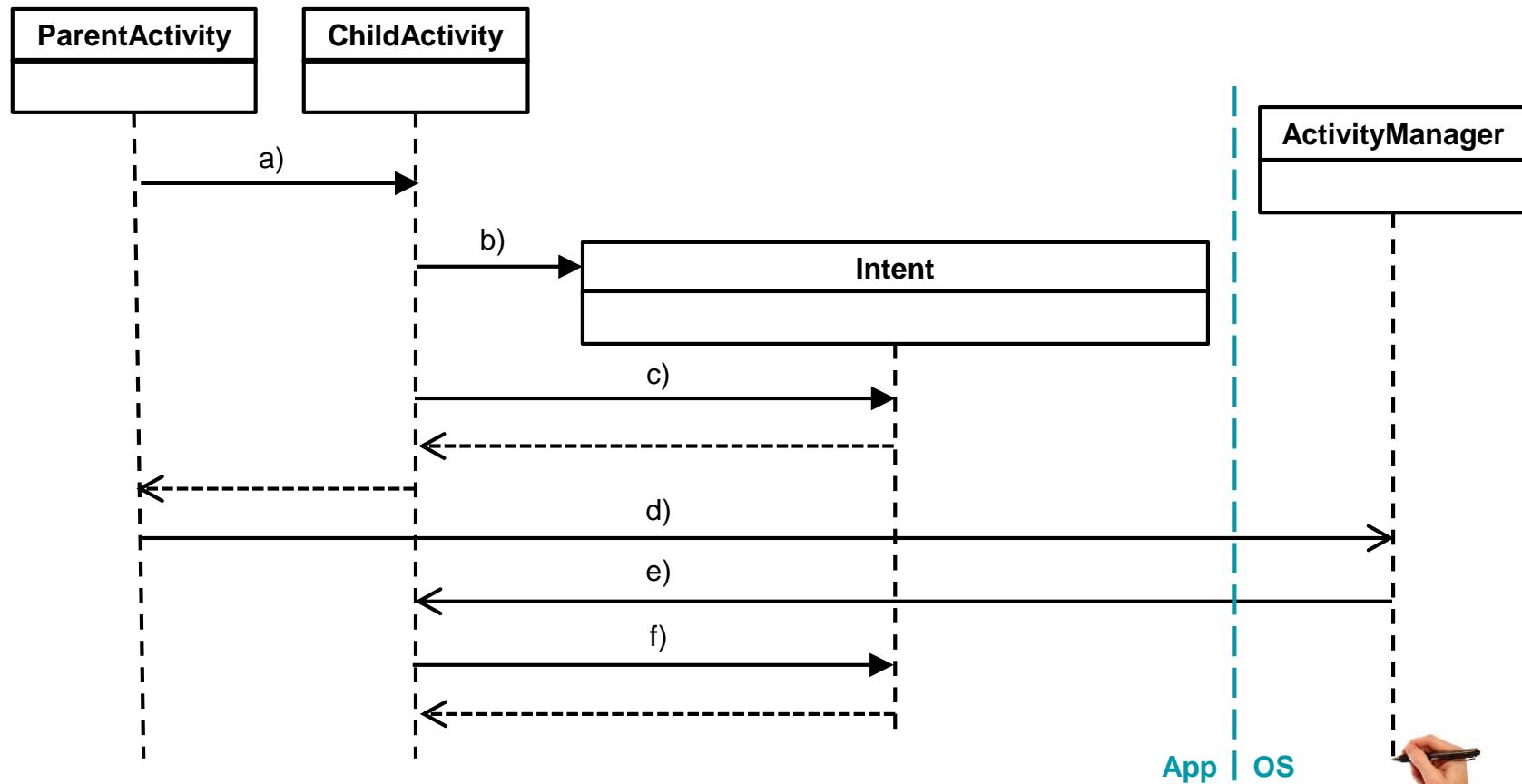
The Activity Stack



In-Class Quiz #5: Communicating with Intents

Assign the proper labels
to the messages in the
sequence diagram:

1. `getBooleanExtra(String) : boolean`
2. `Intent(Context, Class)`
3. `newIntent(Context, boolean) : Intent`
4. `onCreate(Bundle) : void`
5. `putExtra(String, boolean) : void`
6. `startActivity(Intent) : void`

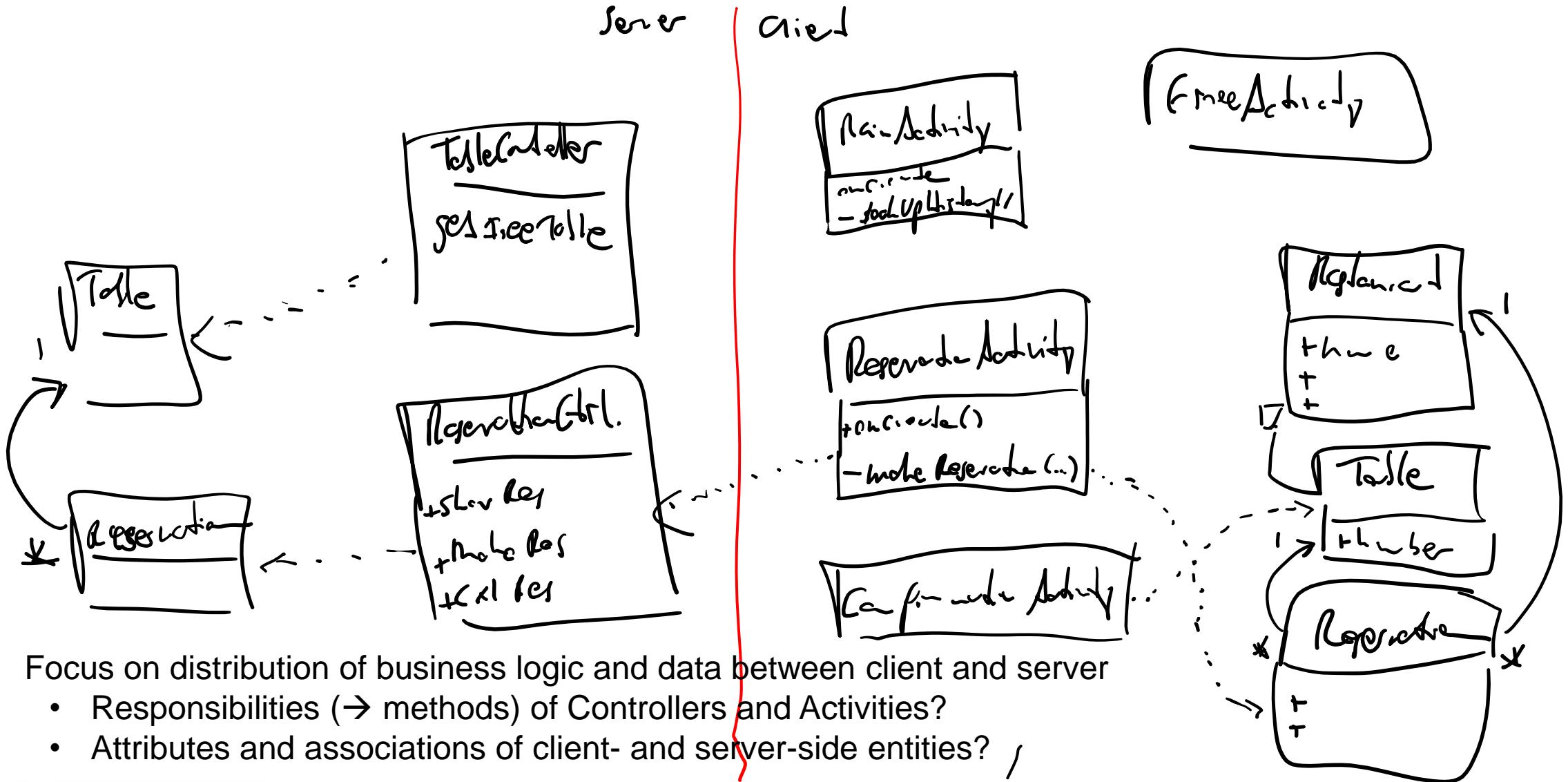


Recap: Assignment 2 – Design Model

- By the deadline specified in your project plan, submit a **design model** in Uglá:
 - UML class diagram of your system (detail level reflecting state of implementation/planning)
 - Clearly distinguish server- and client-side components
 - For client (Android) side, show only Model and Controller classes, not View classes
 - Suitable UML behavioral diagrams to show:
 - User navigation in your app
 - Control flow between key components (**within app, and between client and server**)
- On the Monday after submission, present and **explain** your model to your tutor:
 - How will your system work? What influenced your design choices? What's still unknown?
- **Grading criteria** (25% of this assignment's grade each):
 - System structure is plausible, consistent with requirements and behavioral diagrams
 - User navigation is plausible and shown in a suitable diagram
 - Control flow is plausible and shown in a suitable diagram
 - UML diagrams are clean and syntactically correct



Assignment 2 Example Class Diagram

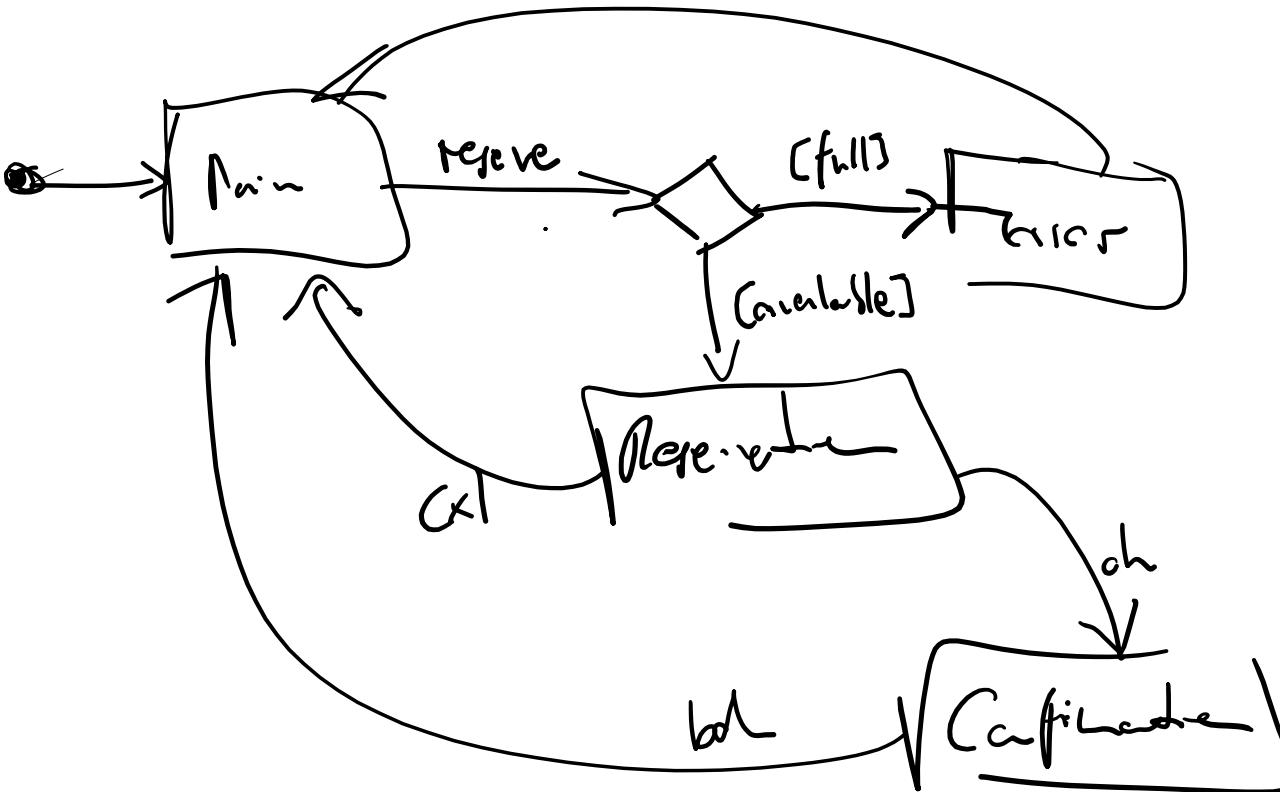


Assignment 2 Example Sequence Diagram



- See HBV601G Quiz 5 for an example of a client-side sequence diagram
- See HBV501G Assignment 2 for an example of a server-side sequence diagram that could be extended with an Android activity contacting a REST controller
- Ensure consistency between diagrams
 - All classes in sequence diagram should also appear in class diagram

Assignment 2 Example State Machine Diagram



- Each screen is a state
- Each click on a widget is a transition
- Conditions can be shown as guards
- Ensure consistency between diagrams
 - All screens in state machine diagram should be activities in class diagram, and vice versa
- Check lecture recording for details

Course Schedule (updated)

| Wk | Thu AM: Lectures | Thu PM: Consultations | Sun: Assignments due |
|----|---------------------------------------|--------------------------|---|
| 1 | Introduction | | |
| 2 | Requirements (<i>video only</i>) | Team & Idea Formation | |
| 3 | Software Estimation | | |
| 4 | Software Estimation | Req. & Plan Draft | #1: Requirements & Project Plan (2 Feb) |
| 5 | Android Basics | Req. & Plan Presentation | |
| 6 | Android Control Flow | | |
| 7 | Android User Interfaces | | #2: OO Design Model (by 1 Mar) |
| 8 | Android Networking | | |
| 9 | Android Storage | | |
| 10 | Android Fragments | | |
| 11 | Software Architecture | | #3: Code Review (in Mar) |
| 12 | Software Architecture | | |
| 13 | Final Exam Prep | | |
| 14 | EASTER BREAK | | |
| 15 | <i>Spare lecture slot (if needed)</i> | Final Presentations | #4: Final Product (19 Apr) |



Hugbúnaðarverkefni 2 / Software Project 2

8. Android Networking and Threading

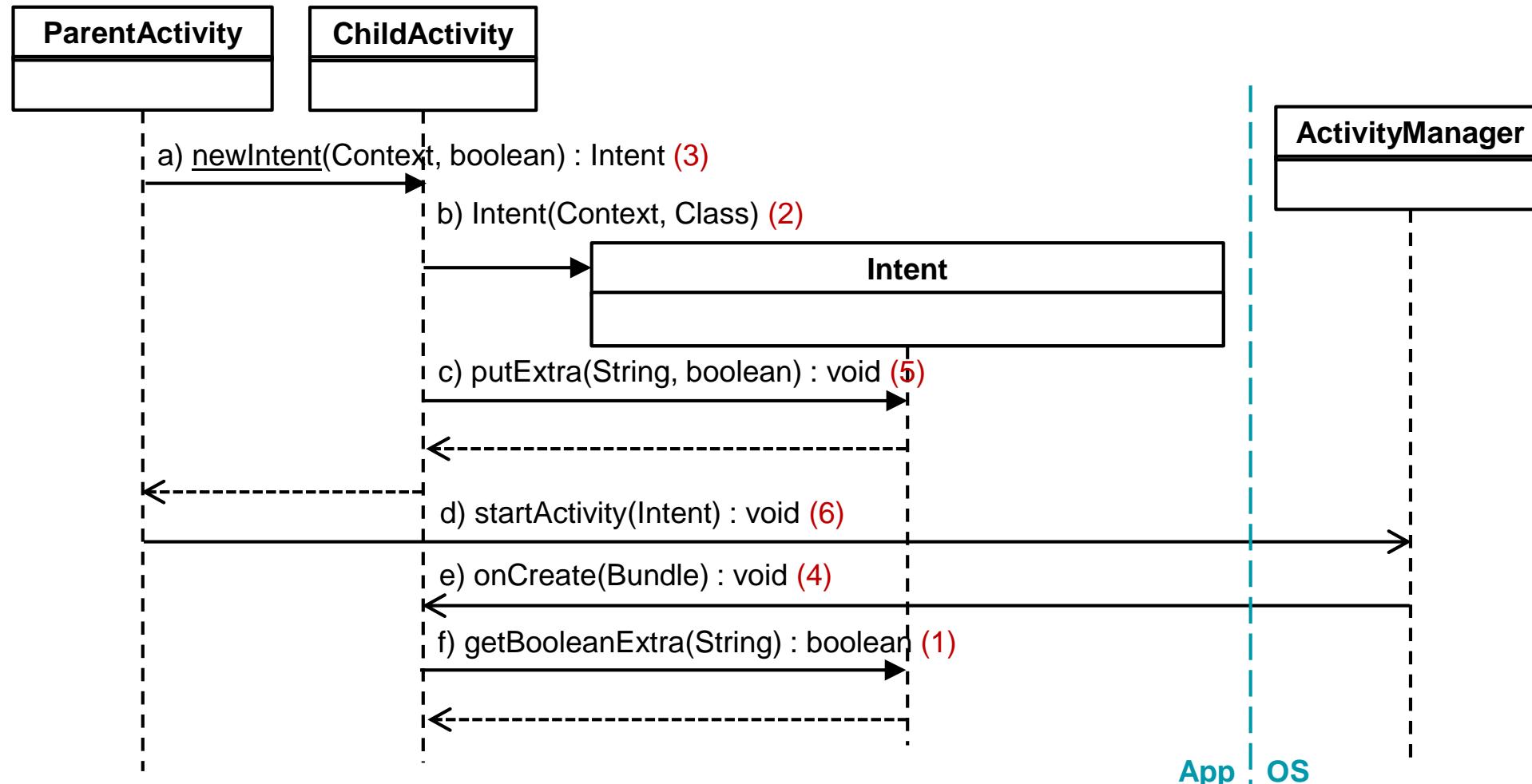
HBV601G – Spring 2020

Matthias Book



HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRVÍSINDASVIÐ
ÍÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD

In-Class Quiz 5 Solution: Communicating with Intents



In-Class Quiz Prep

- Please prepare a small scrap of paper with the following format:

ID: _____ @hi.is Date: _____

a) _____ e) _____
b) _____ f) _____
c) _____ g) _____
d) _____ h) _____

- During class, I'll show you questions that you can answer with some letters
- Hand in your scrap at end of class
- All questions in a quiz have same weight
- All quizzes (8-10 throughout semester) have the same weight
 - Your worst 2 quizzes will be disregarded
- Overall quiz grade counts as optional question worth 7.5% on final exam



Client-Server Communication

using HTTP and JSON

see also:

- Phillips et al.: Android Development, Ch. 23-24 (2nd ed.) / 25-26 (3rd ed.)
- <https://developer.android.com/training/basics/network-ops/connecting.html>



Overview

- The networking part of client-server communication in Android is implemented in the same way as in any other networked application.
 - First part of this class
- Namely, we will use:

```
import java.io.ByteArrayOutputStream;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.URL;
```
- Recommended to review the Java API documentation for these classes
- Things get more complex because of Android-specific multi-threading mechanics though.
 - Second part of this class



Example: A Flickr Viewer



Our goal

Our data source

Explore | Flickr - Photo Sharing!

https://www.flickr.com/explore

flickr You Explore Create

Photos, people, or groups

Explore Trending NEW

More

Explore

Mountain peak, Beach, Flowers, Lake, City skyline, Forest

Prep: Flickr API Documentation

- Flickr provides a rich set of API methods to retrieve data from their site
 - www.flickr.com/services/api
- You'll need an API key to access the Flickr API...
- ...and familiarize yourself with their REST interface

The screenshot shows a web browser window displaying the Flickr API documentation at <https://www.flickr.com/services/api/>. The page has a dark header with the Flickr logo and navigation links for 'You', 'Explore', and 'Create'. A search bar and user profile icons are also present.

The main content area is titled 'The App Garden' and includes links to 'Create an App', 'API Documentation', 'Feeds', and 'What is the App Garden?'. It states that the Flickr API is available for non-commercial use by outside developers. Below this, a sidebar lists 'Read these first:' with links to 'Developer Guide', 'Overview', 'Encoding', 'User Authentication', 'Dates', 'Tags', 'URLs', 'Buddyicons', and 'Flickr APIs Terms of Use'. A red arrow points from the '...and familiarize yourself with their REST interface' bullet point to the 'Request Formats' section.

The 'Request Formats' section is highlighted with a red circle around the 'REST' link, and a red arrow points from the '...and familiarize yourself with their REST interface' bullet point to it. This section also lists 'XML-RPC' and 'SOAP'.

The right side of the page is a list of 'API Methods' grouped by category:

- activity**: flickr.activity.userComments, flickr.activity.userPhotos
- auth**: flickr.auth.checkToken, flickr.auth.getFrob, flickr.auth.getFullToken, flickr.auth.getToken
- auth.oauth**: flickr.auth.oauth.checkToken, flickr.auth.oauth.getAccessToken
- blogs**: flickr.blogs.getList, flickr.blogs.getServices, flickr.blogs.postPhoto
- cameras**: flickr.cameras.getBrandModels, flickr.cameras.getBrands
- collections**: flickr.collections.getInfo, flickr.collections.getTree
- commons**: flickr.common.getInstitutions
- contacts**

At the bottom of the page, there is a footer with the text 'Matthias Book: Software Engineering' and a small logo for 'UNIVERSITATIS ISLANDIAE SIGILLUM'.

Prep: Obtaining a Flickr API Key

- Request a non-commercial API key

The screenshot shows a web browser window titled "Creating an app on Flickr" at the URL <https://www.flickr.com/services/apps/create/noncommercial/>. The Flickr logo is at the top left, and a search bar and user profile are at the top right. The main title is "The App Garden". Below it are links for "Create an App", "API Documentation", "Feeds", and "What is the App Garden?". A pink header says "Tell us about your app:". The form fields include:

- Owner:** Matthias Book
- What's the name of your app?**: FlickrFetchr
- What are you building?**: Simple Android app demonstrating use of REST/JSON for educational purposes
(And trust us when we say you can't be detailed enough)
- Agreements:**
 - I acknowledge that Flickr members own all rights to their content, and that it's my responsibility to make sure that my project does not contravene those rights.
 - I agree to comply with the [Flickr API Terms of Use](#).
- Buttons:** SUBMIT or Cancel

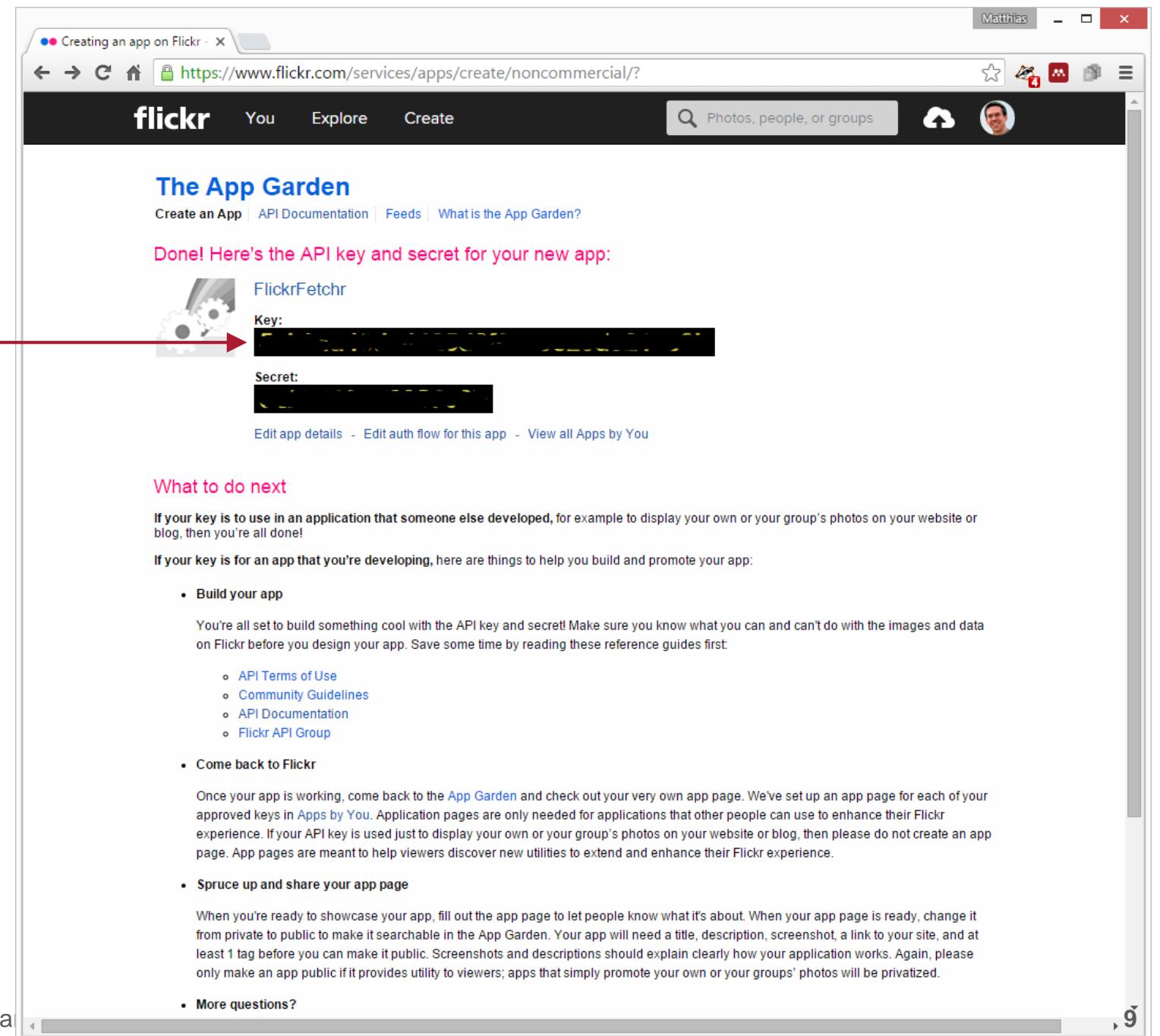
Page Footer:

- About Jobs Blog Mobile Developers Guidelines Feedback Report abuse Help forum English ▾
- Privacy Terms Yahoo Safely Help Flickr, a Yahoo company
- Social media icons: t f tw g+ 8



Prep: Obtaining a Flickr API Key

- You'll just need the key for this project, not the secret



The screenshot shows a web browser window titled "Creating an app on Flickr". The URL is <https://www.flickr.com/services/apps/create/noncommercial/>. The page is titled "The App Garden" and displays a newly created app named "FlickrFetchr". A red arrow points from the text "You'll just need the key for this project, not the secret" to the "Key:" field, which contains a long string of characters. Below the key is a "Secret:" field, also containing a long string of characters. The page includes links for "Edit app details", "Edit auth flow for this app", and "View all Apps by You".

The App Garden

Create an App | API Documentation | Feeds | What is the App Garden?

Done! Here's the API key and secret for your new app:

FlickrFetchr

Key:  [REDACTED]

Secret: [REDACTED]

Edit app details - Edit auth flow for this app - View all Apps by You

What to do next

If your key is to use in an application that someone else developed, for example to display your own or your group's photos on your website or blog, then you're all done!

If your key is for an app that you're developing, here are things to help you build and promote your app:

- Build your app

You're all set to build something cool with the API key and secret! Make sure you know what you can and can't do with the images and data on Flickr before you design your app. Save some time by reading these reference guides first:

- API Terms of Use
- Community Guidelines
- API Documentation
- Flickr API Group

- Come back to Flickr

Once your app is working, come back to the [App Garden](#) and check out your very own app page. We've set up an app page for each of your approved keys in [Apps by You](#). Application pages are only needed for applications that other people can use to enhance their Flickr experience. If your API key is used just to display your own or your group's photos on your website or blog, then please do not create an app page. App pages are meant to help viewers discover new utilities to extend and enhance their Flickr experience.

- Spruce up and share your app page

When you're ready to showcase your app, fill out the app page to let people know what it's about. When your app page is ready, change it from private to public to make it searchable in the App Garden. Your app will need a title, description, screenshot, a link to your site, and at least 1 tag before you can make it public. Screenshots and descriptions should explain clearly how your application works. Again, please only make an app public if it provides utility to viewers; apps that simply promote your own or your groups' photos will be privatized.

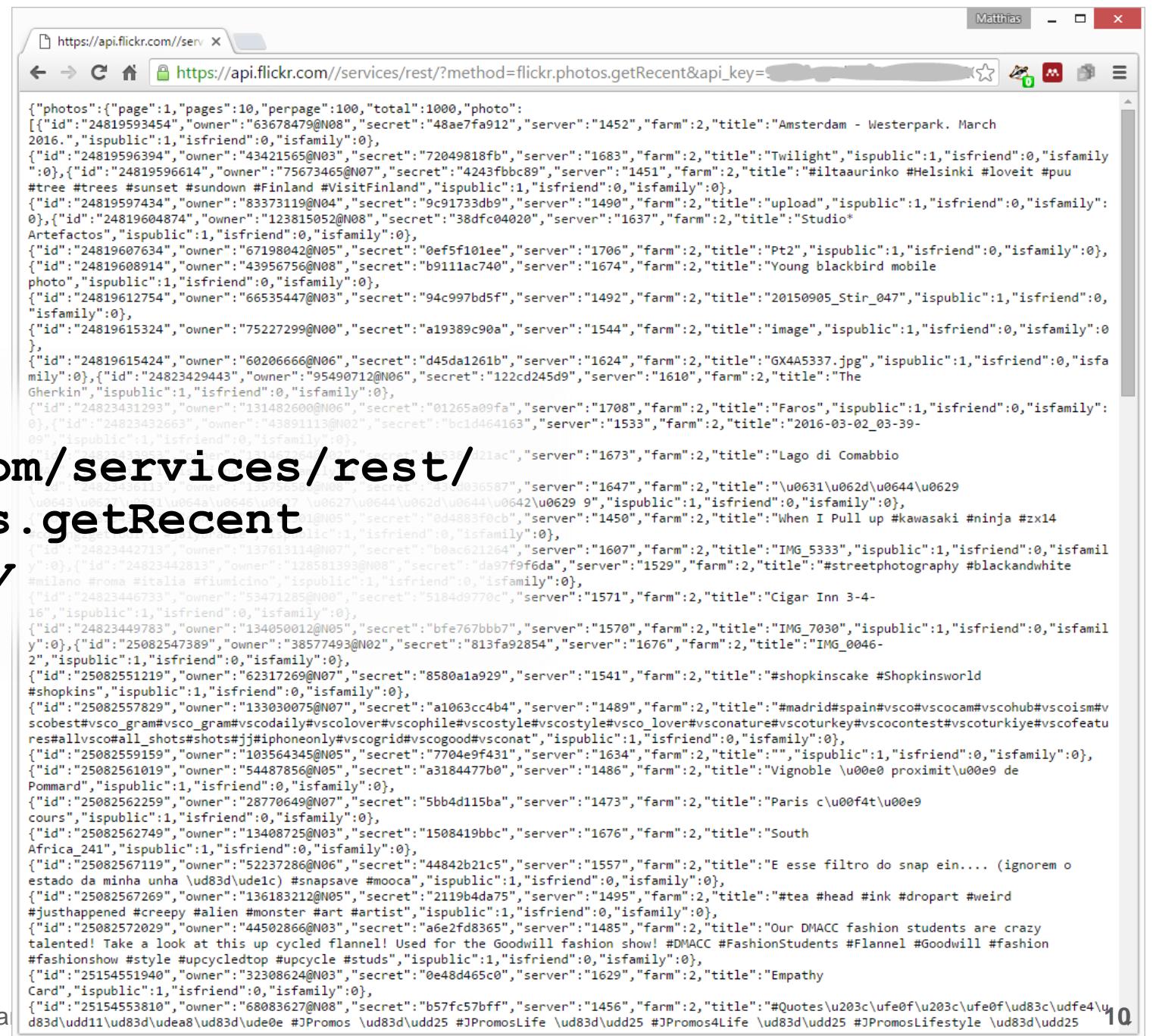
- More questions?



Raw JSON Output of the Flickr API

- Example: Retrieve list of recently uploaded pictures in JSON format

- `https://api.flickr.com/services/rest/?method=flickr.photos.getRecent&api_key=your_API_key&format=json&nojsoncallback=1`
- This is the source format our app will work with.



The screenshot shows a browser window with the URL `https://api.flickr.com//services/rest/?method=flickr.photos.getRecent&api_key=...`. The page displays a large block of JSON data representing a list of recently uploaded photos. The JSON structure includes a "photos" object with a "page" field (1), a "pages" field (10), a "perpage" field (100), and a "total" field (1000). Each photo entry contains fields such as "id", "owner", "secret", "server", "farm", "title", and various metadata like "ispublic" and "isfriend". The JSON is very long, listing numerous photos with unique IDs and titles.



Getting Permission to Network

- Network access from an Android app requires user permission
- This can be requested at the time of installation by including the following lines in `/app/src/main/AndroidManifest.xml`:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.bignerdranch.android.photogallery" >

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application
        ...
        <application />
    </manifest>
```



Reading Any Data via HTTP (FlickrFetcher.java)

```
public class FlickrFetcher {  
    private static final String TAG = "FlickrFetcher";  
    private static final String API_KEY = "your_API_key";  
    public byte[] getUrlBytes(String urlSpec) throws IOException {  
        URL url = new URL(urlSpec);  
        HttpURLConnection conn = (HttpURLConnection)url.openConnection();  
        try {  
            ByteArrayOutputStream out = new ByteArrayOutputStream();  
            InputStream in = conn.getInputStream();  
            if (conn.getResponseCode() != HttpURLConnection.HTTP_OK) {  
                throw new IOException(conn.getResponseMessage() + ": with " + urlSpec);  
            }  
            int bytesRead = 0;  
            byte[] buffer = new byte[1024];  
            while ((bytesRead = in.read(buffer)) > 0) {  
                out.write(buffer, 0, bytesRead);  
            }  
            out.close();  
            return out.toByteArray();  
        } finally { conn.disconnect(); }  
    }  
    public String getUrlString(String urlSpec) throws IOException {  
        return new String(getUrlBytes(urlSpec));  
    }  
    // ...
```

Check request success

Open HTTP connection to service at given URL

Prepare to store data

Request data from URL

Read 1024-byte chunks from HTTP response and write them into output stream

Create byte array from output stream contents

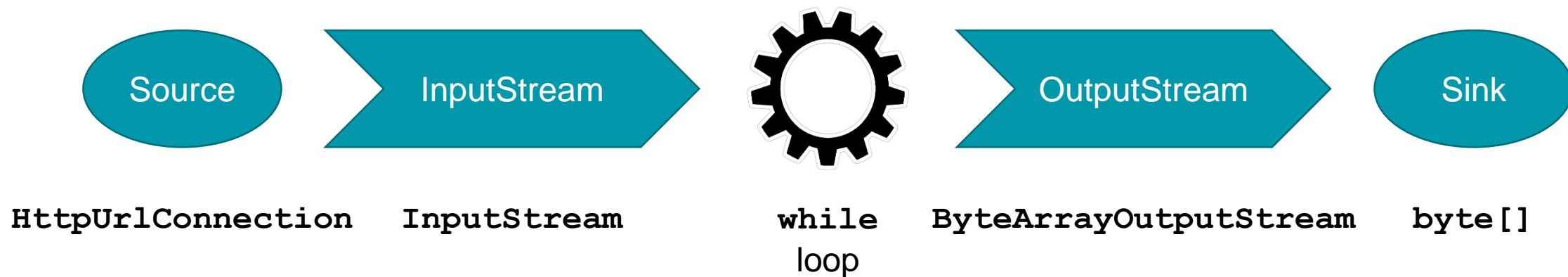
Close network connection, whatever happens

Interpret contents of the byte array as a String

Nothing specific here – can use this in any app



Excursion: Working with Streams



- Java class library provides many more stream types for various operations...
 - Reading from / writing to files, network connections, threads...
 - Reading / writing bytes, strings, objects...
 - Buffering, formatting, encrypting / decrypting, compressing / decompressing...
- ...which can be combined arbitrarily!
- See <https://docs.oracle.com/javase/tutorial/essential/iostreams.html>
- and the Java API documentation for the various subclasses of **java.io.InputStream** and **java.io.OutputStream**

Fetching the List of Recent Pictures from Flickr (FlickrFetchr.java)

```
public List<GalleryItem> fetchItems() {  
    List<GalleryItem> items = new ArrayList<>();  
    try {  
        String url = Uri.parse("https://api.flickr.com/services/rest/").  
            buildUpon()  
            .appendQueryParameter("method", "flickr.photos.getRecent")  
            .appendQueryParameter("api_key", API_KEY)  
            .appendQueryParameter("format", "json")  
            .appendQueryParameter("nojsoncallback", "1")  
            .appendQueryParameter("extras", "url_s")  
            .build().toString();  
        String jsonString = getUrlString(url);  
        Log.i(TAG, "Received JSON: " + jsonString);  
        JSONObject jsonBody = new JSONObject(jsonString);  
        parseItems(items, jsonBody);  
    } catch (IOException ioe) {  
        Log.e(TAG, "Failed to fetch items", ioe);  
    } catch (JSONException je) {  
        Log.e(TAG, "Failed to parse JSON", je);  
    }  
    return items;  
}
```

Data structure in which we will collect the fetched data

Construct a URL with query parameters

Call our previously shown method to retrieve a response string from a URL

Pass the string to an object that provides methods for handling JSON data

A method we'll implement to parse the JSON array and create a list of `GalleryItem` objects from it

The Photo Gallery's Data Model (GalleryItem.java)

- This is the destination format for the data our app receives from the server:

```
public class GalleryItem {  
  
    private String mCaption;  
    private String mId;  
    private String mUrl;  
  
    public String getCaption() { return mCaption; }  
    public void setCaption(String caption) { mCaption = caption; }  
  
    public String getId() { return mId; }  
    public void setId(String id) { mId = id; }  
  
    public String getUrl() { return mUrl; }  
    public void setUrl(String url) { mUrl = url; }  
  
    @Override  
    public String toString() { return mCaption; }  
}
```

Of the data provided in the JSON response, we'll store just the caption, URL and Flickr ID of each image



Parsing JSON Data into Model Objects (FlickrFetchr.java)

```
private void parseItems(List<GalleryItem> items, JSONObject jsonBody)
    throws IOException, JSONException {
    JSONObject photosJsonObject = jsonBody.getJSONObject("photos");
    JSONArray photoJsonArray = photosJsonObject.getJSONArray("photo");

    for (int i = 0; i < photoJsonArray.length(); i++) {
        JSONObject photoJsonObject = photoJsonArray.getJSONObject(i);

        GalleryItem item = new GalleryItem();
        item.setId(photoJsonObject.getString("id"));
        item.setCaption(photoJsonObject.getString("title"));
        if (!photoJsonObject.has("url_s")) {
            continue;
        }
        item.setUrl(photoJsonObject.getString("url_s"));

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

Obtain the JSON array holding photo references

For each JSON array element...

...construct a new `GalleryItem`, populate its attributes...

...and add it to the list of `items`



Retrieving Bitmaps via HTTP (ThumbnailDownloader.java)

- Retrieving the image files that the `GalleryItems` refer to is quite easy:
 1. We use the method `byte[] getUrlBytes(String url)` that we implemented in `FlickrFetchr` to retrieve the binary data for the file (e.g. a JPG image)
 2. We use the static method `Bitmap decodeByteArray (byte[] data, int offset, int length)` of `android.graphics.BitmapFactory` to convert the image data into a `Bitmap` object that can be incorporated into our user interface.
- Example:

```
byte[] bitmapBytes = new FlickrFetchr().getUrlBytes(url);
final Bitmap bitmap = BitmapFactory.decodeByteArray(bitmapBytes, 0, bitmapBytes.length);
```
- Transfer-wise, this is all we need to retrieve and parse JSON and binary data...
- ...but when using these methods in a mobile app, we still need to consider that network communication is *slow*.

Android Threading

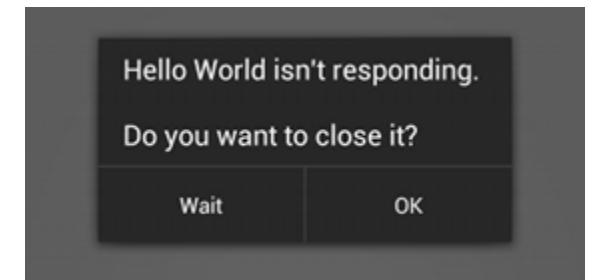
see also:

- <https://developer.android.com/training/articles/perf-anr.html>
- <https://developer.android.com/guide/components/processes-and-threads>



The Main/UI Thread

- The Activities of each Android app are executed in the app's **main thread**
 - The main thread (also called **UI thread**) is an infinite loop that
 - waits for events from the user or system (e.g. Activity startups, button taps etc.)
 - and calls the respective event handlers
 - If execution of any event handler takes too long, the UI becomes unresponsive
 - If an app doesn't respond to a UI event within 5 seconds, the Android OS will offer to close it
- Anything that has the potential to take a long time, e.g...
- any network operations
 - file operations that may be blocked due to concurrent access
 - high-complexity or high-volume algorithms
- ...must be executed in a different thread than the main thread
- Android prevents network operations on main thread (**NetworkOnMainThreadException**)



“App not responding”
(ANR) dialog

Using Worker Threads Instead of the UI Thread

- Any operations that are not instantaneous must not be performed on the main UI thread, but separate “background” or “worker” threads.
- Simplest solution: Start a new thread to perform the time-consuming work:

```
public void onClick(View v) {  
    new Thread(new Runnable() {  
        public void run() {  
            // implementation of a potentially time-consuming task  
            // ...  
        }  
    }).start();  
}
```

- Problem: We cannot update the UI from any other thread than the UI thread

Updating the UI from Within a Worker Thread

- Solution: Calling `Activity.runOnUiThread(Runnable)` allows executing operations on the UI thread from within another thread:

```
public void onClick(View v) {  
    new Thread(new Runnable() {  
        public void run() {  
            // implementation of a potentially time-consuming task  
            // ...  
            EnclosingActivityName.this.runOnUiThread(new Runnable() {  
                public void run() {  
                    // implementation of a UI update  
                }  
            });  
        }  
    }).start();  
}
```

“`this`” alone would refer to the anonymous
Runnable, so we need to prefix it with the name of the
activity class in which this event handler is defined

- This construct however gets unwieldy for more complex interaction with the worker thread.

Background Threads

- There are several mechanisms for executing work in background threads (i.e. other threads than the app's UI thread):
 - Android provides one background thread for “light”, short-lived, infrequent tasks tied to a particular activity (so called **AsyncTasks**) (\rightarrow next slides; *Android textbook, Ch. 23/25 – deprecated*)
 - All **AsyncTasks** are executed sequentially in the app's same background thread
 - [Dis]advantage: You cannot/need not manage the tasks' threading mechanics
 - You can create your own background threads for “heavy”, long-running tasks that are tied to a particular activity (so called **HandlerThreads**) (\rightarrow *Android textbook, Ch. 24*)
 - Each **HandlerThread** is executed in a background thread of its own, controlled by an activity
 - [Dis]advantage: You can/need to manage the threading mechanics yourself
 - You can create your own background threads for any tasks that are executed independently from activities (so called **IntentServices**) (\rightarrow end of this class; *Android textbook, Ch. 26*)
 - Each **IntentService** is executed in a thread of its own, started via an intent
 - Advantages: Completely independent from an app's UI; can be scheduled

Background Tasks

using `AsyncTask`

see also:

- Phillips et al.: Android Development, Ch. 23 (2nd ed.) / 25 (3rd ed.)
- <https://developer.android.com/reference/android/os/AsyncTask.html>



Invocation of an AsyncTask (PhotoGalleryFragment.java)

```
public class PhotoGalleryFragment extends Fragment {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setRetainInstance(true);  
        AsyncTask task = new FetchItemsTask();  
        task.execute();  
        // ...  
    }  
  
    // ...
```

Keep this fragment object intact even if its hosting activity is re-created (e.g. due to device rotation)

So we don't need to re-create the **AsyncTask**

Create the new task and tell the background thread to execute it

Note: **AsyncTasks** in background thread are queued, so execution may not begin immediately!



Implementation of an AsyncTask (PhotoGalleryFragment.java)

```
public class PhotoGalleryFragment extends Fragment {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setRetainInstance(true);  
        AsyncTask task = new FetchItemsTask();  
        task.execute();  
        // ...  
    }  
    // ...  
  
    private class FetchItemsTask extends AsyncTask<Void,Void,List<GalleryItem>> {  
        @Override  
        protected List<GalleryItem> doInBackground(Void... params) {  
            return new FlickrFetcher().fetchItems();  
        }  
    }  
}
```

Make this an
AsyncTask

Type of
input data

Type of
output data

Q: Where does `doInBackground`'s
return value end up?

The `task.execute` call is in the
wrong place and in a different thread 😞

In this case, calling our method
that retrieves the JSON string
from Flickr's REST API

`doInBackground` contains
the code to be executed in
the background thread

Returning Data from an AsyncTask to the Main Thread (PhotoGalleryFragment.java)

```
public class PhotoGalleryFragment extends Fragment {  
  
    private List<GalleryItem> mItems = new ArrayList<>();  
  
    // ...  
  
    private class FetchItemsTask extends AsyncTask<Void, Void, List<GalleryItem> > {  
  
        @Override  
        protected List<GalleryItem> doInBackground(Void... params) {  
            return new FlickrFetcher().fetchItems();  
        }  
  
        @Override  
        protected void onPostExecute(List<GalleryItem> items) {  
            mItems = items;  
            setupAdapter();  
        }  
    }  
}
```

A: `doInBackground`'s return value is received by `AsyncTask`'s private implementation and passed to the tasks's `onPostExecute` method...

...which is executed in the *main* instead of the *background* thread!

Copy the return value to the fragment's member variable

Takes care of displaying the `GalleryItems`

For clarity, we skip all aspects of the View layer in this example – see Ch. 9, 23 & 24 of the Android textbook and corresponding source code for details.

Optional: Progress Updates from Within an AsyncTask

```
ProgressBar mProgressBar = (ProgressBar) findViewById(R.id.progress_bar);  
// ...  
  
AsyncTask<Void, Integer, Void> task = new AsyncTask<Void, Integer, Void>() {  
  
    public Void doInBackground(Void... params) {  
        Integer percentComplete;  
        while (...) { // some loop condition  
            // ...background operations...  
            publishProgress(percentComplete);  
        }  
    }  
  
    public void onProgressUpdate(Integer... params) {  
        int progress = params[0];  
        mProgressBar.setProgress(progress);  
    }  
};  
// ...  
task.execute();
```

Type of progress data

Keep track of progress (here: in `percentComplete`) in the `doInBackground` method (executed in background thread)

Call `publishProgress` whenever the progress indicator in the View shall be updated

...where a suitable widget can be updated

A call to `publishProgress` will trigger execution of `onProgressUpdate` in the main thread...

Cancelling an AsyncTask

- If you need to cancel an **AsyncTask** before it has begun or completed, you can call its `cancel(boolean mayInterrupt)` method. Then:
 - If the cancelled task's `doInBackground` method hasn't been called yet, it won't be called
 - If the `doInBackground` method is already active: If `mayInterrupt` is
 - `false`: the method will keep running, but the task's "cancelled" flag will be set
 - You can check this flag occasionally in your `doInBackground` implementation by calling the `boolean isCancelled()` method, and elect to finish work prematurely if you find the flag set
 - `true`: the method will be interrupted immediately – try to avoid this
 - After cancelling an **AsyncTask**, its `onCancelled` method will be called instead of its `onPostExecute` method
- When and where cancellation of an **AsyncTask** makes sense may vary:
 - Possibly cancel it in the activity's/fragment's `onStop` or `onDestroy` method
 - Possibly simply let it run to completion
 - But ensure it doesn't invoke methods on an activity that may already be in an invalid state

Background Services

using IntentService

see also:

- Phillips et al.: Android Development, Ch. 26 (2nd ed.) / 28 (3rd ed.)
- <https://developer.android.com/guide/components/services.html>



Activities vs. Services

- So far, we know only activities as the prime components of our Android apps
 - An **IntentService** is a component treated in some ways like an activity, i.e...
 - it can be started and destroyed
 - it is invoked via an intent
 - ...but differing in some important ways from activities, namely:
 - it has no associated user interface (view)
 - it runs in its own (background) thread
- This means that unlike **AsyncTasks** and **HandlerThreads**, **IntentServices** can perform background work that is completely independent from activities:
- A service can be active even when none of the app's activities are alive
 - A service can be automatically (e.g. periodically) invoked by the Android OS

Implementing an IntentService's Business Logic (PollService.java)

```
public class PollService extends IntentService {  
  
    public static Intent newIntent(Context context) {  
        return new Intent(context, PollService.class);  
    }  
  
    @Override  
    protected void onHandleIntent(Intent intent) {  
        if (!isNetworkAvailableAndConnected()) { return; }  
        // ...perform the business logic...  
    }  
  
    private boolean isNetworkAvailableAndConnected() {  
        ConnectivityManager cm = (ConnectivityManager) getSystemService(CONNECTIVITY_SERVICE);  
        boolean isNetworkAvailable = cm.getActiveNetworkInfo() != null;  
        boolean isNetworkConnected = isNetworkAvailable &&  
            cm.getActiveNetworkInfo().isConnected();  
        return isNetworkConnected;  
    }  
}
```

Make this an IntentService

Factory method for an intent (“command”) that other components can use to invoke this service

Core method implementing the business logic that shall be executed in the background

If the business logic requires network access, check if we actually *can* network first

Recommended to use this code in your own apps too

Network needs to be available (not the case if user disabled network access for background tasks) and connected



An IntentService's Life: Reacting to Commands

- An intent addressed at an **IntentService** is called a **command**.
- Upon the first command, the **IntentService** starts up and puts the command in the queue.
- Further commands are added to the queue in the order they arrive.
- Meanwhile, the **IntentService** takes one command after another from the queue and calls **onHandleIntent** for each of it.
- When the queue is empty, the **IntentService** is destroyed.
- When new commands arrive, a new instance of the **IntentService** is created, and its lifecycle starts from the beginning.



Necessary Declarations in the Manifest (AndroidManifest.xml)

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.bignerdranch.android.photogallery" >

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <application
        android:allowBackup="true" android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name" android:theme="@style/AppTheme" >
        <activity
            android:name=".PhotoGalleryActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service android:name=".PollService" />
    </application>
</manifest>
```

Services need to be declared just like activities



Immediate Invocation of an IntentService

- An **IntentService** can be invoked immediately by sending it an intent (i.e. a command):

```
Intent i = PollService.newIntent(getActivity());  
getActivity().startService(i);
```

`getActivity` is required to obtain a `Context` when calling from within a fragment (*covered later*). From within an activity, you could just use `this`.

- This code could be used by some method of an activity or fragment anytime they need the **IntentService** to do something.
 - Note the syntax is quite similar to how you would send an intent to an activity.
 - The behavior is quite different though: Instead of transitioning to a new activity...
- If the **IntentService** doesn't exist yet, it will be started in the background.
- If the **IntentService** is already running, the command will be added to its queue and executed when all previous commands have been handled

Delayed Invocation of an IntentService

- An **IntentService** can be invoked automatically by the Android OS even when none of your activities are alive, i.e. your app's process is shut down
- Using an **AlarmManager**, you can let this happen periodically or at a specific point in time.
 - Upon each scheduled **alarm**, a predefined intent will be fired to invoke your service
- **Power consumption considerations**
 - Caution: When creating periodic alarms, consider that executing an **IntentService** more frequently will drain the battery faster
 - Android implements several strategies to conserve power:
 - The interval between alarms must be at least 60 seconds
 - Alarms will be “bundled” so several services can be invoked at the same time, even if that means that the intervals between alarms won't have exactly the specified length
 - When the device is sleeping (screen off), due alarms will not be created (so as not to wake the device up for a service invocation), unless you explicitly specify otherwise

Scheduling Alarms with an AlarmManager (PollService.java)

```
public class PollService extends IntentService {  
    // ...  
  
    public static void setServiceAlarm(Context context, boolean isOn) {  
        Intent i = PollService.newIntent(context);  
        PendingIntent pi = PendingIntent.getService(context, 0, i, 0);  
  
        AlarmManager alarmManager = (AlarmManager)  
            context.getSystemService(Context.ALARM_SERVICE);  
  
        if (isOn) {  
            alarmManager.setInexactRepeating(AlarmManager.ELAPSED_REALTIME,  
                SystemClock.elapsedRealtime(), AlarmManager.INTERVAL_FIFTEEN_MINUTES, pi);  
        } else {  
            alarmManager.cancel(pi);  
            pi.cancel();  
        }  
    }  
}
```

Typically invoked by some activity or fragment

Create an intent that should later be used to invoke our service

Wrap intent *i* into a PendingIntent

Obtain the system's AlarmManager

Create an alarm (i.e. fire the PendingIntent) approx. every 15 min.

Only one alarm schedule per PendingIntent!



Inexact Scheduling of Alarms with an AlarmManager

- One-time and repeating alarms are scheduled with **AlarmManager**'s methods

```
void set           (int type, long trigger,           PendingIntent operation)
void setInexactRepeating(int type, long trigger, long interval, PendingIntent operation)
```

- **trigger** indicates the time of the first alarm

- For **type==ELAPSED_REALTIME**, milliseconds (ms) since booting the device (incl. sleep)
 - For **type==RTC**, a particular time (in ms) – Caution: UTC timezone, not adjusted for locale!

- **interval** indicates the interval between subsequent alarms

- Use one of provided **INTERVAL_** constants (e.g. **_HOUR**) or a custom value (in ms)

- Note: Scheduling is inexact in order to conserve battery!

- First alarm will not go off before scheduled trigger, but may go off sometime later
 - Next alarms will not go off before expiry of interval, but may go off almost a full interval later
 - Alarms will not go off if device is asleep, but be postponed until other reason for wake-up, unless **_WAKEUP** is appended to **type** constant (e.g. **RTC_WAKEUP**)

More Exact Scheduling Mechanisms

- **AlarmManager** also provides mechanisms that suggest exact scheduling (e.g. `setRepeating`), but since API level 19, these also use inexact scheduling.
- For more precise control over when your alarms occur, you can use `setWindow` (`int type, long windowStartMillis, long windowLengthMillis, PendingIntent operation`) to narrow the time window in which the alarm must go off.
- **AlarmManager** is just intended to fire intents. For more lightweight and precise timing operations (ticks, timeouts etc.), **Handler** is easier and more efficient.



Checking if an Alarm is Scheduled for a PendingIntent (PollService.java)

```
public static boolean isServiceAlarmOn(Context context) {  
    Intent i = PollService.newIntent(context);  
    PendingIntent pi = PendingIntent.  
        getService(context, 0, i, PendingIntent.FLAG_NO_CREATE);  
    return pi != null;  
}
```

Indicates if we have done this before, i.e. if we have scheduled an alarm

Try to wrap the intent in a **PendingIntent** again (remember: only possible once)

Return the **PendingIntent** if it already exists, or **null** if it doesn't exist

Default behavior (parameter 0) would be:
Return the **PendingIntent** if it already exists, or create it if it doesn't



Interacting with the User from Within a Service

- Since services do not have associated View components (i.e. no UI), the user will (and should) normally not notice that they are active.
- When a service wants to make itself aware to the user, it has two options:
 - Send an intent that will invoke an activity, which will come to the foreground
 - Create a notification that will appear in the notifications drawer (accessible by swiping down from top of screen)
 - Often, notifications are wired up such that tapping on the notification will fire an intent that invokes a corresponding activity



Sending a Notification from a Service (PollService.java)

```
@Override  
protected void onHandleIntent(Intent intent) {  
    // ...business logic...  
  
    Resources resources = getResources();  
    Intent i = PhotoGalleryActivity.newIntent(this);  
    PendingIntent pi = PendingIntent.getActivity(this, 0, i, 0);  
  
    Notification notification = new NotificationCompat.Builder(this)  
        .setTicker(resources.getString(R.string.new_pictures_title))  
        .setSmallIcon(android.R.drawable.ic_menu_report_image)  
        .setContentTitle(resources.getString(R.string.new_pictures_title))  
        .setContentText(resources.getString(R.string.new_pictures_text))  
        .setContentIntent(pi)  
        .setAutoCancel(true)  
        .build();  
  
    NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);  
    notificationManager.notify(0, notification);  
  
    // ...business logic...
```

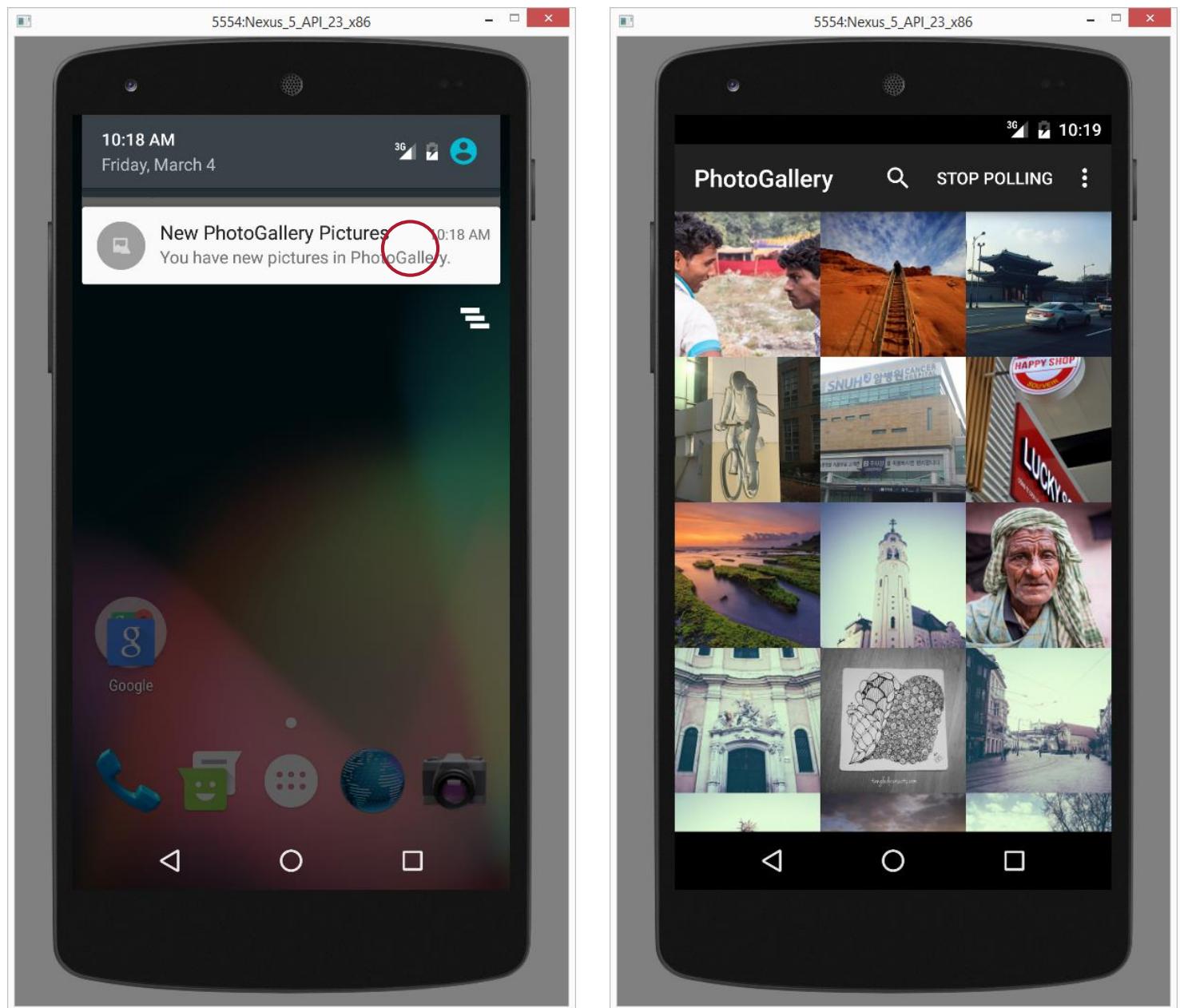
Create and wrap an intent that will invoke `PhotoGalleryActivity` upon tapping on the notification

Create a notification with the given text, icon, intent to be fired upon tap (see API doc for details)

Delete notification from drawer when user taps on it

Post the notification

Running the App



Summary: Background Threads

- There are several mechanisms for executing work in background threads (i.e. other threads than the app's UI thread):
 - Android provides one background thread for “light”, short-lived, infrequent tasks tied to a particular activity (so called **AsyncTasks**) (\rightarrow *Android textbook, Ch. 23/25, deprecated*)
 - All **AsyncTasks** are executed sequentially in the app's same background thread
 - (Dis)advantage: You cannot/need not manage the tasks' threading mechanics
 - You can create your own background threads for “heavy”, long-running tasks that are tied to a particular activity (so called **HandlerThreads**) (\rightarrow *Android textbook, Ch. 24*)
 - Each **HandlerThread** is executed in a background thread of its own, controlled by an activity
 - (Dis)advantage: You can/need to manage the threading mechanics yourself
 - You can create your own background threads for any tasks that are executed independently from activities (so called **IntentServices**) (\rightarrow *end of this class; Android textbook, Ch. 26/28*)
 - Each **IntentService** is executed in a thread of its own, started via an intent
 - Advantages: Completely independent from an app's UI, can be scheduled

In-Class Quiz 6: Threading



- Note whether the following properties apply to **(A)syncTasks, (I)ntentServices, (H)andlerThreads** (*note that some properties may apply to several threading mechanisms*):
 - a) Executed in app's background thread
 - b) Executed in background thread of its own
 - c) Scheduling controlled by Android OS
 - d) Scheduling controlled by developer
 - e) Started via intent
 - f) Started via method invocation
 - g) Independent of particular activities
 - h) Tied to particular activity



Hugbúnaðarverkefni 2 / Software Project 2

9. Data Storage in Android

HBV601G – Spring 2020

Matthias Book



HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRVÍSINDASVIÐ
ÍÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD

In-Class Quiz 6 Solution: Android Threading



- Note whether the following properties apply to
(A)syncTasks, (I)ntentServices, (H)andlerThreads
(note that some properties may apply to several threading mechanisms):
- a) Executed in app's background thread A
- b) Executed in background thread of its own I, H
- c) Scheduling controlled by Android OS A, [I]
- d) Scheduling controlled by developer I, H
- e) Started via intent I
- f) Started via method invocation A, H
- g) Independent of particular activities I
- h) Tied to particular activity A, H

In-Class Quiz Prep

- Please prepare a small scrap of paper with the following format:

ID: _____ @hi.is Date: _____

a) _____ e) _____
b) _____ f) _____
c) _____ g) _____
d) _____ h) _____

- During class, I'll show you questions that you can answer with some letters
- Hand in your scrap at end of class
- All questions in a quiz have same weight
- All quizzes (8-10 throughout semester) have the same weight
 - Your worst 2 quizzes will be disregarded
- Overall quiz grade counts as optional question worth 7.5% on final exam



Overview: Where to Store Your App's Data?

- **savedInstanceState** bundle is only appropriate for short-term data saving while an activity is inactive, and may be destroyed at any time to free memory

More persistent alternatives:

- Static media for integration in your app's user interface (e.g. images)
 - Resource directory of your project
- Persistent storage of simple data that only your app works with (e.g. game state)
 - **SharedPreferences** file with key/value map in your app's private internal storage directory
- Persistent storage of large data that only your app works with (e.g. cached data)
 - Files in your app's private internal storage directory
- Dynamic, unstructured data that you exchange with other apps (e.g. documents)
 - Files in the user's external storage directories
- Dynamic, structured data that only your app works with (e.g. a to-do list)
 - Local SQLite database in your app's internal storage directory
- Data that you want to share with other users (e.g. messages)
 - Central data storage on a remote server



Resources

see also:

- Phillips et al.: Android Development, end of Ch. 2
- <http://developer.android.com/guide/topics/resources/providing-resources.html>



Resource Directories

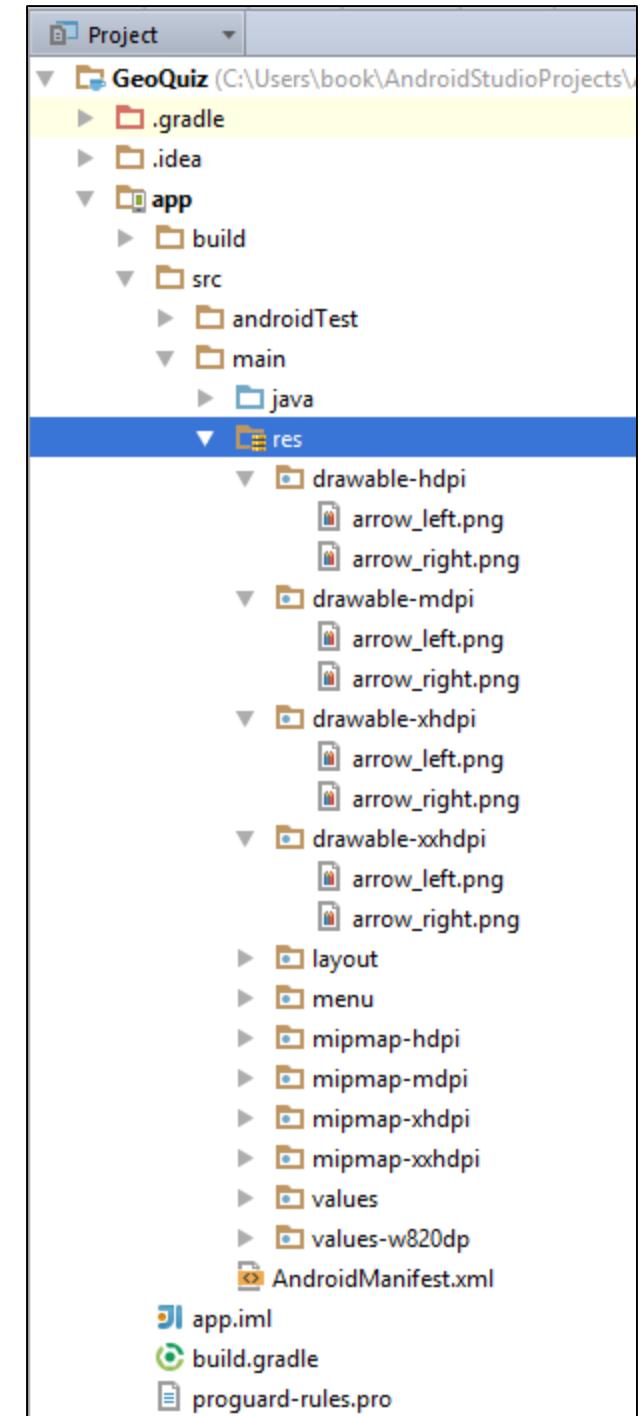
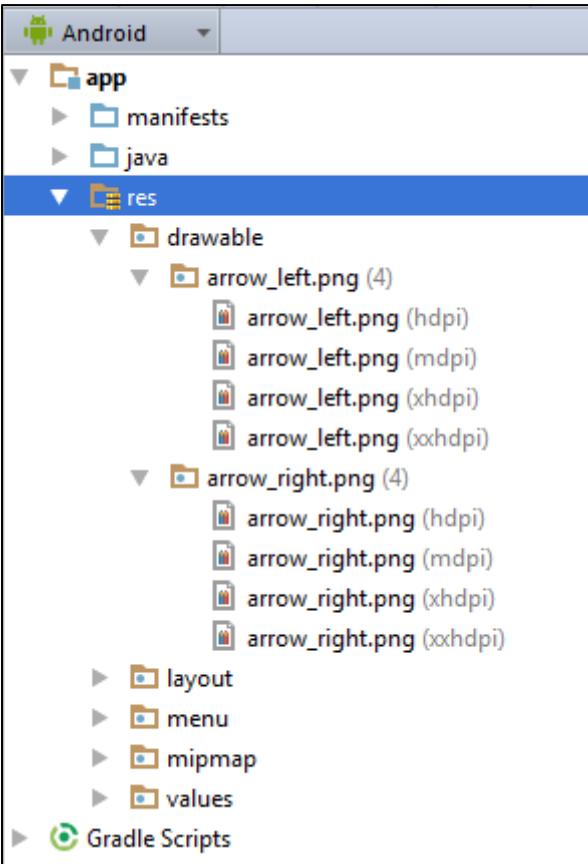
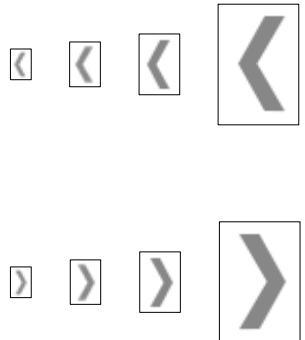
- Recap: A resource is any static piece of your app that is not code
 - i.e. an image, sound or XML file
 - Resources are static files, i.e. delivered with your app at deployment and not changed
- Resources are stored in subfolders of your project's **app/res** directory, e.g.
 - **animator/** for XML files describing animations
 - **color/** for XML files describing color states
 - **drawable/** for PNG/JPG/GIF bitmap graphics or XML files describing vector graphics
 - **layout/** for XML files describing view layouts
 - **menu/** for XML files describing app menus
 - **raw/** for arbitrary static files in their original format
 - **values/** for XML files describing simple values such as strings
 - **xml/** for XML files containing arbitrary data (e.g. app-specific configurations)

Configuration Qualifiers for Alternative Resources

- Recap: In different device configurations (i.e. orientation or language), your app may require different resources, e.g.
 - different layout files for portrait or landscape orientation
 - different string files for different languages
 - different image files for different screen resolutions
- Recap: “Configuration qualifiers” are suffixes added to resource directories to indicate in which configuration the resources should be used, e.g.
 - `layout/` for portrait layout, `layout-land/` for landscape layout
- For images, you need to provide different versions for different resolutions:
 - `drawable-mdpi/`, `drawable-hdpi/`, `drawable-xhdpi/`, `drawable-xxhdpi/` for screen pixel densities of 160, 240, 320 and 480 dpi, respectively
- Android will pick the most appropriate image at runtime and scale it as needed

Providing Alternative Resources

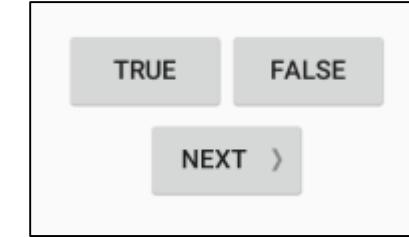
- Example directory structure:



Accessing Resources

- Resource IDs are automatically created from resources' filenames and can be accessed in XML and Java
- Example: Adding the image `arrow_right.png` to a button

```
<Button  
    android:id="@+id/next_button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="@string/next_button"  
    android:drawableRight="@drawable/arrow_right"  
    android:drawablePadding="4dp"/>
```



- In Java, a resource's ID can be obtained through the `R` class
 - e.g. `R.drawable.arrow_right` (Note: this is an `int` ID value, not the actual image)
- Note: No configuration qualifier is included in the reference!
 - Android automatically chooses the resource from the appropriate directory for the current device configuration

Shared Preferences

see also:

- <https://developer.android.com/training/data-storage/shared-preferences>



Storing App State vs. Activity State

- A **savedInstanceState** bundle is a volatile, activity-specific data storage
 - Can only be accessed by the activity that created it
 - May be destroyed by Android at any time to free up memory
 - Only suitable for maintaining activity state through a destroy-recreate lifecycle transition
- For reliable storage of small pieces of app state...
 - that shall be accessible by different activities of your app
 - that shall be guaranteed to persist independently of any activity's lifecycle
 - that shall be deleted only upon uninstalling your app
- ...use **SharedPreferences** instead.
- **SharedPreferences** are simple key/value pairs residing in a persistent file that is accessible only by your app.

Obtaining a SharedPreferences File

- To obtain a **SharedPreferences** key/value map for your app, call the `SharedPreferences getSharedPreferences(String name, int mode)` method of any **Context** (i.e. **Activity**) of your app.
 - name** is a filename of your choice (in case you want to maintain several shared pref. files)
 - mode** must be **MODE_PRIVATE** (all other options are deprecated)
- Example:

```
Context context = getActivity();
SharedPreferences sharedPref = context.getSharedPreferences(
    getString(R.string.preference_file_key), Context.MODE_PRIVATE);
```

Obtaining the context e.g.
from within a fragment

SharedPreferences are not
readable by any other apps

It's good practice to use string constants
everywhere to avoid bugs by typos.
Since a shared preference is not restricted to
one activity, we may not want to declare the
string in a Java constant in an activity, but
declare it in the app's strings.xml file instead.

Reading Key/Value Pairs from SharedPreferences

- The **SharedPreferences** instance we obtained provides a variety of methods for reading key/value pairs of different types
 - All require the **key** to look up, as well as a **default** value to return if the key wasn't found
 - e.g. `int getInt(String key, int default)`
 - similar for other primitive types (**boolean**, **float**, etc.) and **String**
- Example:

```
int highScore = sharedPref.getInt(getString(R.string.saved_high_score), 0);
```

Writing Key/Value Pairs to SharedPreferences

- To write data, we first need to call the `edit()` method on the `SharedPreferences` we obtained previously
- This will give us a `SharedPreferences.Editor` that provides methods for writing and manipulating key/value pairs of different types
 - e.g. `.putInt(String key, int value)` to set/update the `value` for the `key`
 - similar for `boolean`, `float`, `String` etc.
 - `remove(String key)` to remove the value for the `key`
- Finally, call `apply()` to write all changes to the persistent file (important!)
 - The `edit`, `put` and `remove` methods all return `Editor` instances to enable call chaining
- Example:

```
sharedPref.edit()  
    .putInt(getString(R.string.saved_high_score), newHighScore)  
    .putFloat(getString(R.string.saved_balance), newBalance)  
    .apply();
```

File Storage

see also:

- <https://developer.android.com/training/data-storage>
- <https://developer.android.com/training/data-storage/app-specific>
- Phillips et al.: Android Development, beginning of Ch. 14 (internal)
- Phillips et al.: Android Development, middle of Ch. 16 (external)



Internal vs. External Storage

Internal Storage

- Built-in, always available
- Files are accessible only by your app
- Does not require special permission
- All files deleted upon uninstalling app

➤ **Suitable for files that should be access-restricted**

- e.g. app state, sensitive data

External Storage

- Might be SD card unmounted by user
 - highly unlikely in modern devices
- Files accessible by all of user's apps
- Requires permission upon installation
- “Private” files deleted upon uninstalling app, “public” files remain

➤ **Suitable for files that are intended for sharing/outliving your app**

- e.g. created media/documents



Files in Internal Storage

- Every app has a “sandbox” directory that is only accessible by the app itself
 - Not by other apps, and not by the user (unless the device is rooted or in an emulator)
- Located in the device’s `/data/data/<app package>` directory
 - e.g. `/data/data/is.hi.hbv601g.geoquiz`
- **Context** (i.e. **Activity**) objects provide methods for internal storage access:
 - `File getFilesDir()` returns handle of the app’s internal storage directory
 - `FileInputStream openFileInput(String name)` opens existing file for reading
 - `FileOutputStream openFileOutput(String name, int mode)` opens file for writing, creating it if necessary
 - `File getDir(String name, int mode)` returns handle of a subdirectory of the app’s internal storage directory, creating it if necessary
 - `String[] fileList()` returns list of file names in app’s internal storage directory
- Use the Java class library’s standard `java.io` classes for file and stream ops

<https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/io/package-summary.html>



Example: Writing a File to Internal Storage

```
String filename = "myfile";
String s = "Hello world!";
FileOutputStream outputStream;

try {
    outputStream = openFileOutput(filename, Context.MODE_PRIVATE);
    outputStream.write(s.getBytes());
    outputStream.close();
} catch (Exception e) {
    e.printStackTrace();
}
```

Create file in the app's internal storage directory and get a stream to write into it

MODE_PRIVATE: recreate file even if it exists
MODE_APPEND: if file exists, append to it, otherwise create it

Write bytes into the file



Cache Directory in Internal Storage

- Besides files that are intended for persistent local storage, an app may choose to store data on the device for caching purposes
 - e.g. to avoid having to repeatedly request it from a remote server
- Cached data is special insofar as
 - it typically gets stale (and thus useless) after some time
 - it is not an authoritative information source
 - i.e. if it is stale or missing, the original source can simply be queried again
- To reflect these properties, cache data should be stored in a special directory:
 - `File getCacheDir()` returns handle of the app's cache directory in internal storage
- Android can then delete cache data automatically if file storage space runs low
 - But there is no guarantee of deletion (except upon uninstalling the app)
 - App should ensure the cache directory does not grow massively

Location of Internal Storage Directories

- In Android Studio, open View > Tool Windows > Device File Explorer to look at emulated device's file system

- /data/data** contains

- sandbox directories
for various apps

- (only visible on rooted or emulated devices)

Device File Explorer

Emulator Pixel_2_API_26 Android 8.0.0, API 26

| Name | Permissions | Date | Size |
|------------------------------------|-------------|------------------|------|
| acct | dr-xr-xr-x | 2019-02-04 17:44 | 0 B |
| cache | drwxrwx--- | 2019-02-04 17:44 | 4 KB |
| config | drwxr-xr-x | 2019-02-04 17:44 | 0 B |
| d | lrwxrwxrwx | 1970-01-01 00:00 | 17 B |
| data | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| app | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| data | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| android | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.backupconfirm | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.bips | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.bookmarkprovider | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.calculator2 | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.callogbackup | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.camera2 | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.captiveportallogin | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.carrierconfig | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.cellbroadcastreceiver | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.certinstaller | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.chrome | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.companiondevicemanager | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.contacts | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.cts.ctsshim | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.cts.priv.ctsshim | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.customlocale2 | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.defcontainer | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| com.android.development | drwxrwx--x | 2019-02-04 17:46 | 4 KB |

Files in External Storage

- External storage used to be a removable SD card in early devices, but is today usually emulated by just another folder in the device's built-in flash memory
 - This is so-called “primary” external storage
 - Today's devices can have “additional” external storage in the form of a removable SD card
- Still, it's good practice to check whether external storage is available before attempting to use it:
 - `Environment.getExternalStorageState()` should return `MEDIA_MOUNTED`



Private vs. Public External Storage

- “Private” directories in external storage
 - **File** handle for directory obtained through `getExternalFilesDir(String type)` instance method of a **Context** object (typically, an **Activity**)
 - Files created here are readable/writable by other apps, but your app is in charge of the folder
 - This could e.g. be data that you publish for consumption by other apps, or vice versa
 - The directory is deleted when your app is uninstalled
- “Public” directories in external storage
 - **File** handle for directory obtained through static `getExternalStoragePublicDir(String type)` method of the **Environment** class
 - Files created here are readable/writable by other apps, and the user is in charge of the folder
 - This is typically data “owned” by the user, i.e. the user’s photos, music, documents etc.
 - The directory remains on the device when your app is uninstalled
- Note: On a multi-user device, apps only have access to the external storage of the user they are running as (i.e. one user can’t see another’s photos)

External Storage Directory Types

- The **String type** parameter passed to the previously shown methods indicates the type of file you want to access/store
- This determines which external storage directory will be provided to you, e.g.:
 - **DIRECTORY_PICTURES** for photos
 - **DIRECTORY_MOVIES** for video files
 - **DIRECTORY_MUSIC** for audio files containing music (for use by media players)
 - **DIRECTORY_RINGTONES** for audio files containing ringtones (not for use by media players)
 - **DIRECTORY_DOCUMENTS** for documents
- It's recommended to use the above constants (defined in **Environment**)
 - In private external storage, you could also use your own directory names
 - In public external storage, using your own names is discouraged so you don't clutter up the user's directory structure

Example: Creating a Photo Album

- In private external storage:

```
public File getAlbumStorageDir(Context context, String albumName) {  
    File file = new File(context.getExternalFilesDir(  
        Environment.DIRECTORY_PICTURES), albumName);  
    if (!file.mkdirs()) {  
        Log.e(LOG_TAG, "Directory not created");  
    }  
    return file;  
}
```

Get handle for app's private external storage directory for pictures

- In public external storage:

```
public File getAlbumStorageDir(String albumName) {  
    File file = new File(Environment.getExternalStoragePublicDirectory(  
        Environment.DIRECTORY_PICTURES), albumName);  
    if (!file.mkdirs()) {  
        Log.e(LOG_TAG, "Directory not created");  
    }  
    return file;  
}
```

Try creating the subdirectory **albumName** in there

Get handle for user's public external storage directory for pictures

After this, use the Java class library's standard **java.io** classes for file and stream operations



Location of External Storage Directories

- Primary external storage is emulated on this device
 - i.e. in built-in flash memory, not on a removable SD card
 - Storage for current user
- Private external storage
 - Managed by individual apps
 - Deleted when uninstalled
 - Readable by any app
- Public external storage
 - Managed by the user
 - Outlives any app
 - Structured by file type



| Name | Permissions | Date | Size |
|---|-------------|------------------|-------|
| ► sbin | drwxr-x--- | 1970-01-01 00:00 | 120 B |
| ► sdcard | lrwxrwxrwx | 1970-01-01 00:00 | 21 B |
| ▼ storage | drwxr-xr-x | 2019-02-04 17:47 | 80 B |
| ► emulated | drwx--x--x | 2019-02-04 17:46 | 4 KB |
| ► self | drwxr-xr-x | 2019-02-04 17:44 | 60 B |
| ► primary | lrwxrwxrwx | 2019-02-04 17:44 | 19 B |
| ► Alarms | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| ▼ Android | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| ► data | drwxrwx--x | 2019-02-12 12:32 | 4 KB |
| ► com.android.vending | drwxrwx--x | 2019-02-04 17:53 | 4 KB |
| ► com.google.android.apps.docs | drwxrwx--x | 2019-02-12 12:16 | 4 KB |
| ► com.google.android.apps.maps | drwxrwx--x | 2019-02-04 17:48 | 4 KB |
| ► com.google.android.apps.nexuslauncher | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| ► com.google.android.gms | drwxrwx--x | 2019-02-04 17:47 | 4 KB |
| ► com.google.android.googlequickstep | drwxrwx--x | 2019-02-04 17:49 | 4 KB |
| ► com.google.android.inputmethod | drwxrwx--x | 2019-02-04 17:48 | 4 KB |
| ► com.google.android.music | drwxrwx--x | 2019-02-12 12:30 | 4 KB |
| ► com.google.android.videos | drwxrwx--x | 2019-02-12 12:32 | 4 KB |
| ► com.google.android.youtube | drwxrwx--x | 2019-02-04 17:48 | 4 KB |
| ► DCIM | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| ► Download | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| ► Movies | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| ► Music | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| ► Notifications | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| ► Pictures | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| ► Podcasts | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| ► Ringtones | drwxrwx--x | 2019-02-04 17:46 | 4 KB |
| ► sys | dr-xr-xr-x | 2019-02-04 17:44 | 0 B |
| ► system | drwxr-xr-x | 1970-01-01 00:00 | 4 KB |
| ► var | lrwxrwxrwx | 2019-02-04 17:44 | 9 B |

Obtaining Permission to Access External Storage

- In order to be able to access external storage, your app must obtain permission from the user upon installation
- To ask for the required permission, include one of the following lines in your manifest file (`app/src/main/AndroidManifest.xml`):
 - `<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />`
 - `<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />`
- Write permission implies read permission
- Read permission is currently not required by Android, but may be in later release
 - So if your app needs it, declare it already now so your app will keep working

Local Databases

Using the Room Persistence Library



see also:

- <https://developer.android.com/training/data-storage/room>

SQLite and Room in Android

- Use a **local (client-side) database** for easy access to any structured data that
 - can be kept in its entirety on the client device
 - needs to be accessed only by that device's user
- Use a **server-side database** for any structured data that
 - is shared between users
 - is too large for storage on the device

Technology for client-side database:

- SQLite, a relational database storing data in simple files (www.sqlite.org)
- Room, a persistence library providing an abstraction layer over SQLite
- Android includes the SQLite library in its standard library
- Room can easily be added to an app



Preparation: Adding Architecture Components

- Add Google's Maven repository to your project's `build.gradle` file:
 - i.e. e.g. for GeoQuiz, to `GeoQuiz\build.gradle`

```
allprojects {  
    repositories {  
        google()  
        jcenter()  
    }  
}
```

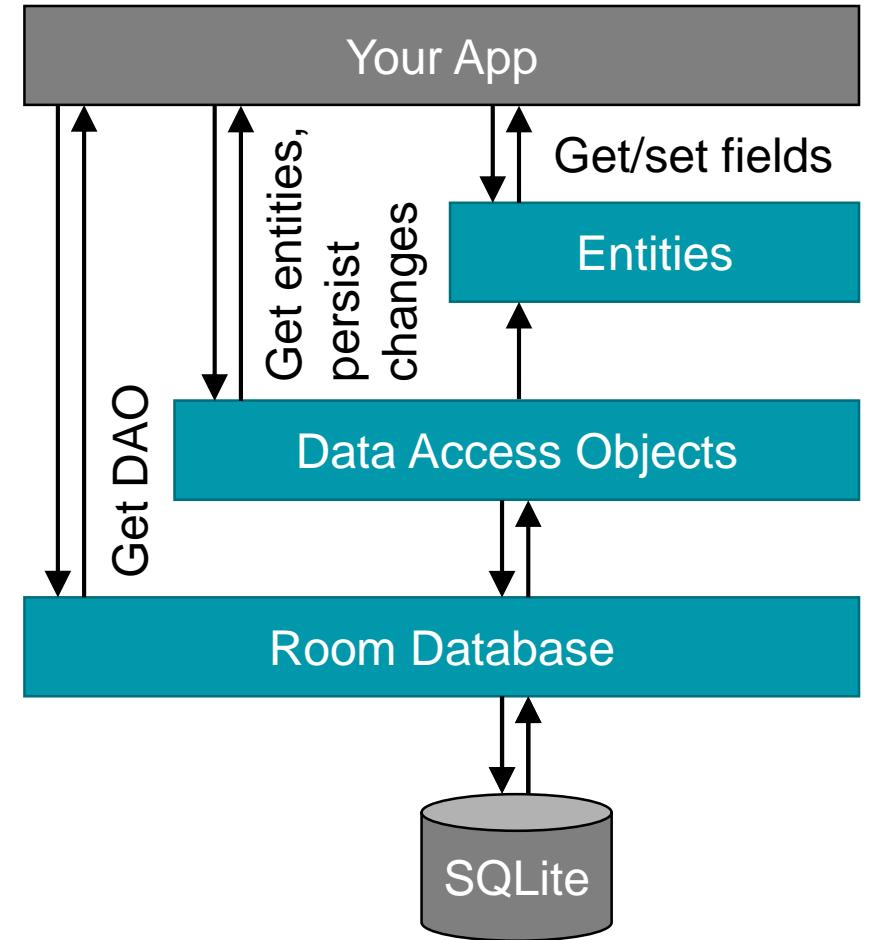
- Add the dependencies for Room to your app's `build.gradle` file:
 - i.e. e.g. for GeoQuiz, to `GeoQuiz\app\build.gradle`

```
dependencies {  
    def room_version = "2.2.4"  
  
    implementation "androidx.room:room-runtime:$room_version"  
    annotationProcessor "androidx.room:room-compiler:$room_version"  
  
    // Test helpers  
    testImplementation "androidx.room:room-testing:$room_version"  
}
```



Room Persistence Library Architecture

- **@Database** provides an abstraction of the underlying SQLite database containing the app's persisted relational data
- **@Dao** instances contain methods for getting entities from the database and saving changes back to the database
- **@Entity** instances represent tables within the database and provide access to its columns



Example: Storing User Objects

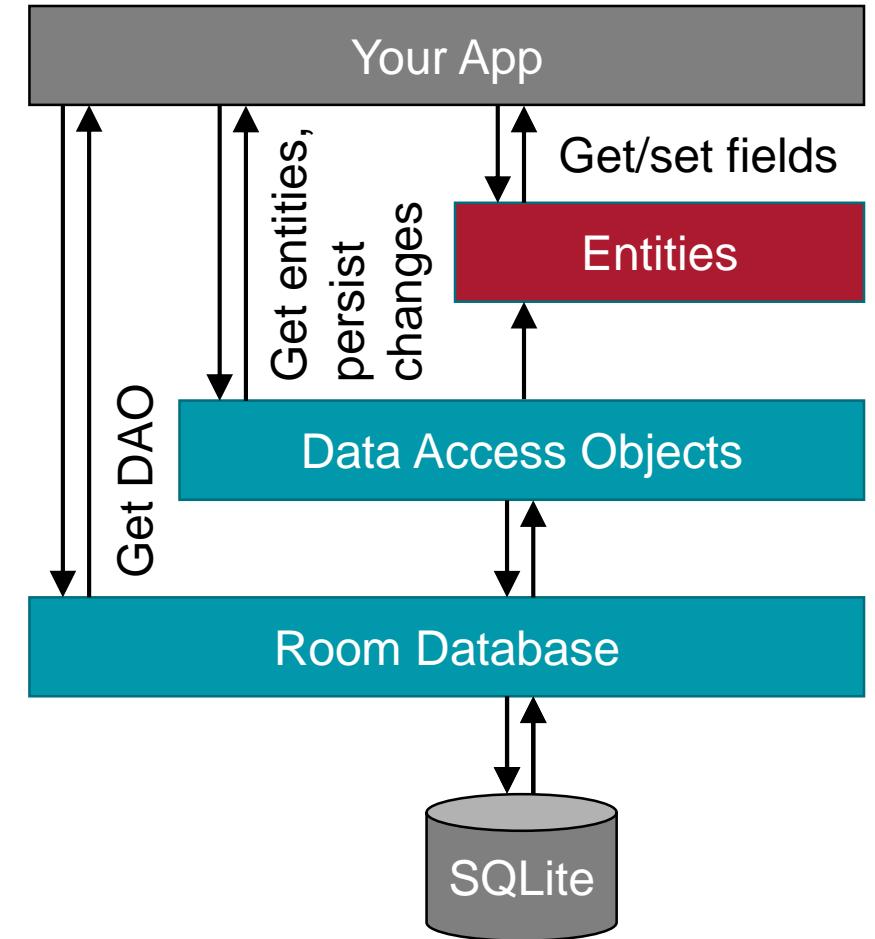
User Entity

```
@Entity  
public class User {  
    @PrimaryKey  
    public int id;  
  
    @ColumnInfo(name = "first_name")  
    public String firstName;  
  
    @ColumnInfo(name = "last_name")  
    public String lastName;  
}
```

Required for at least one attribute

Optional, using attribute name otherwise

- Note: Contrary to other object-relational mappers, Room forbids object references between entities for performance reasons
 - Eager loading can waste memory
 - Lazy loading can cost time at inopportune moments

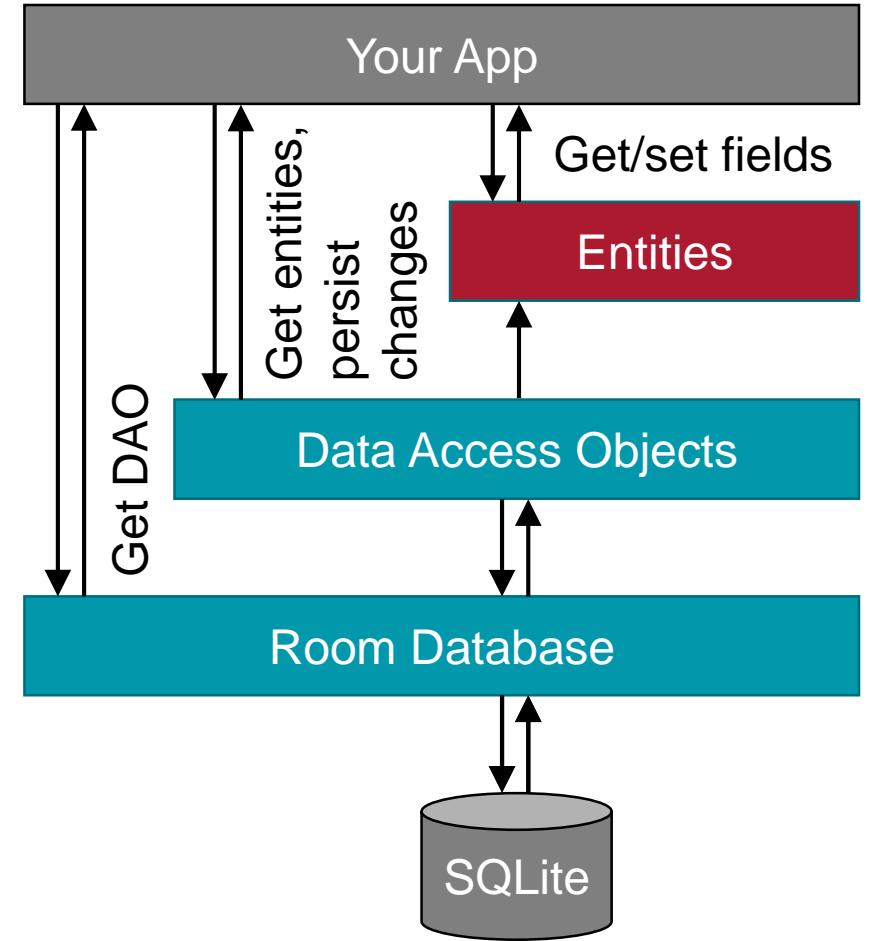


Example: Storing User Objects

Book Entity

```
@Entity(foreignKeys = @ForeignKey(  
    entity = User.class,  
    parentColumns = "id",  
    childColumns = "user_id",  
    onDelete = CASCADE))  
public class Book {  
    @PrimaryKey  
    public int bookId;  
  
    public String title;  
  
    @ColumnInfo(name = "user_id")  
    public int userId;  
}
```

Delete all of a user's books
if the user is deleted



- Solution: Define foreign key relationships
 - To request dependent data yourself when needed
 - To let SQLite take care e.g. of cascading deletions

Example: Storing User Objects

User Data Access Object

```
@Dao
public interface UserDao {
    @Query("SELECT * FROM user")
    List<User> getAll();

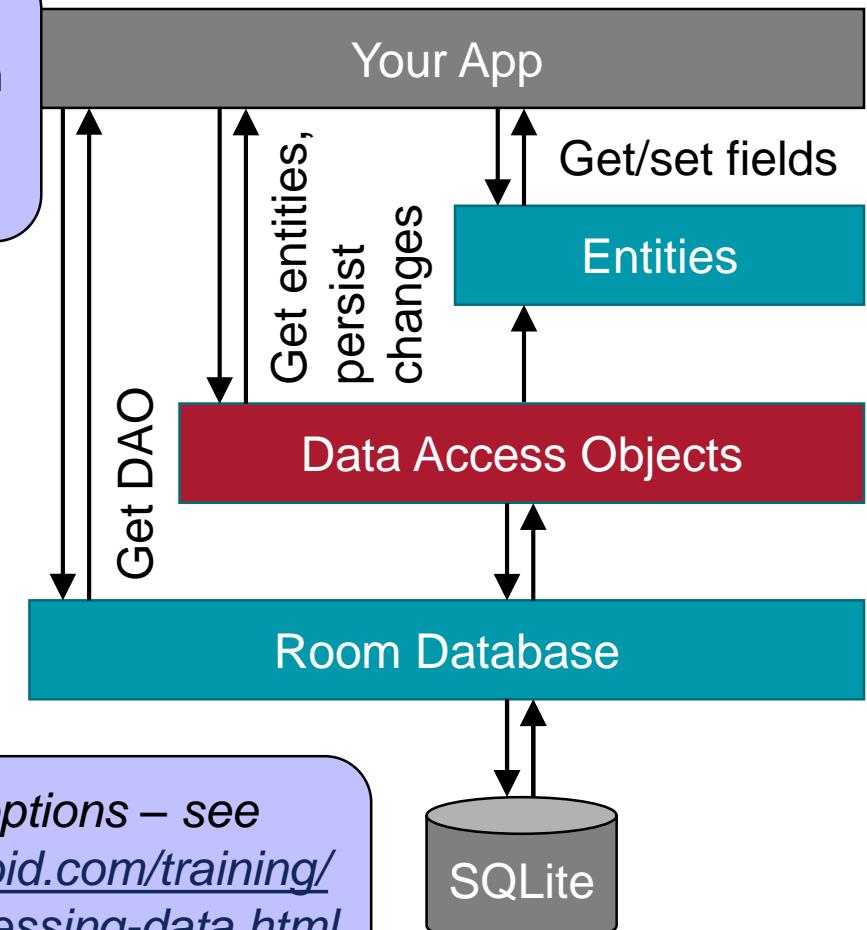
    @Query("SELECT * FROM user " +
           "WHERE id IN (:userIds)")
    List<User> loadAllByIds(int[] userIds);

    @Query("SELECT * FROM user " +
           "WHERE first_name LIKE :first AND " +
           "last_name LIKE :last LIMIT 1")
    User findByName(String first, String last);

    @Insert
    void insertAll(User... users);

    @Delete
    void delete(User user);
}
```

Compile-time error if bind parameter does not match the name of a method parameter



Many more query options – see
<https://developer.android.com/training/data-storage/room/accessing-data.html>
for details

Example: Storing User Objects Database

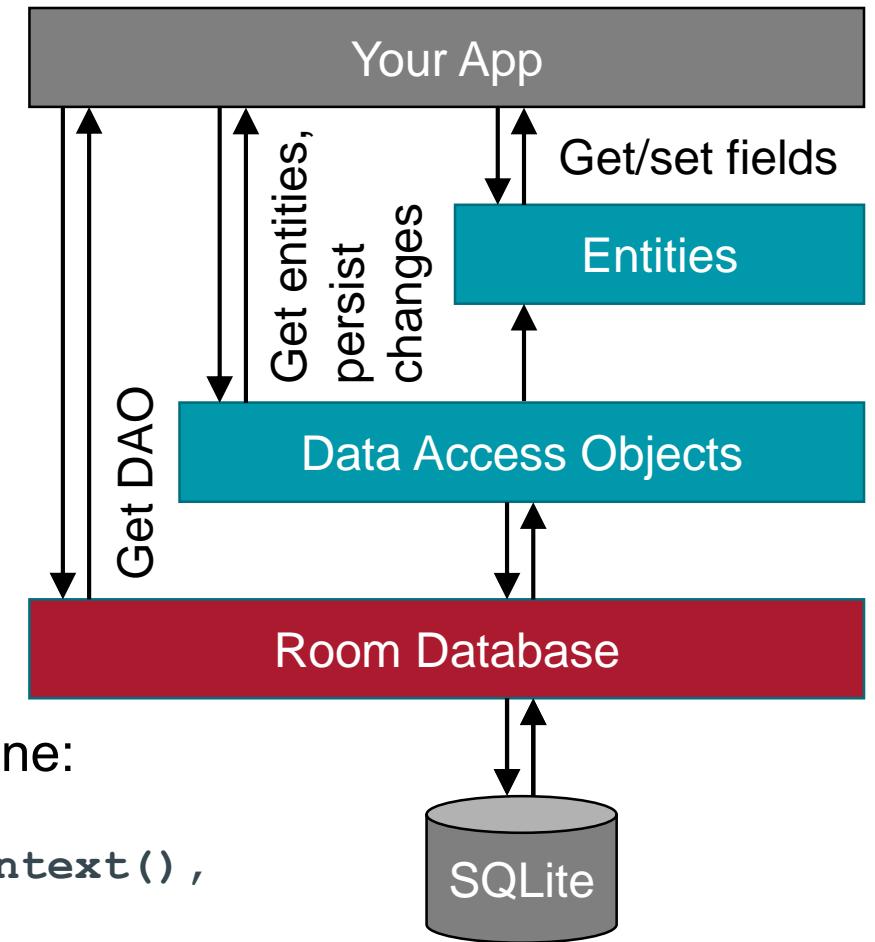
- Declaring the database abstraction:

```
@Database(entities = {User.class}, version = 1)
public abstract class AppDatabase
    extends RoomDatabase {
    public abstract UserDao userDao();
}
```

- Obtaining an instance of the database:

- Note: Wrap this in a singleton pattern since the instance is expensive, and you'll rarely need more than one:

```
AppDatabase db = Room.databaseBuilder(getApplicationContext(),
    AppDatabase.class, "database-name").build();
```



Example: Storing User Objects Accessing Entities via the DAO

- Obtain an instance of the database:

```
AppDatabase db = Room.databaseBuilder(  
    getApplicationContext(),  
    AppDatabase.class, "database-name") .build();
```

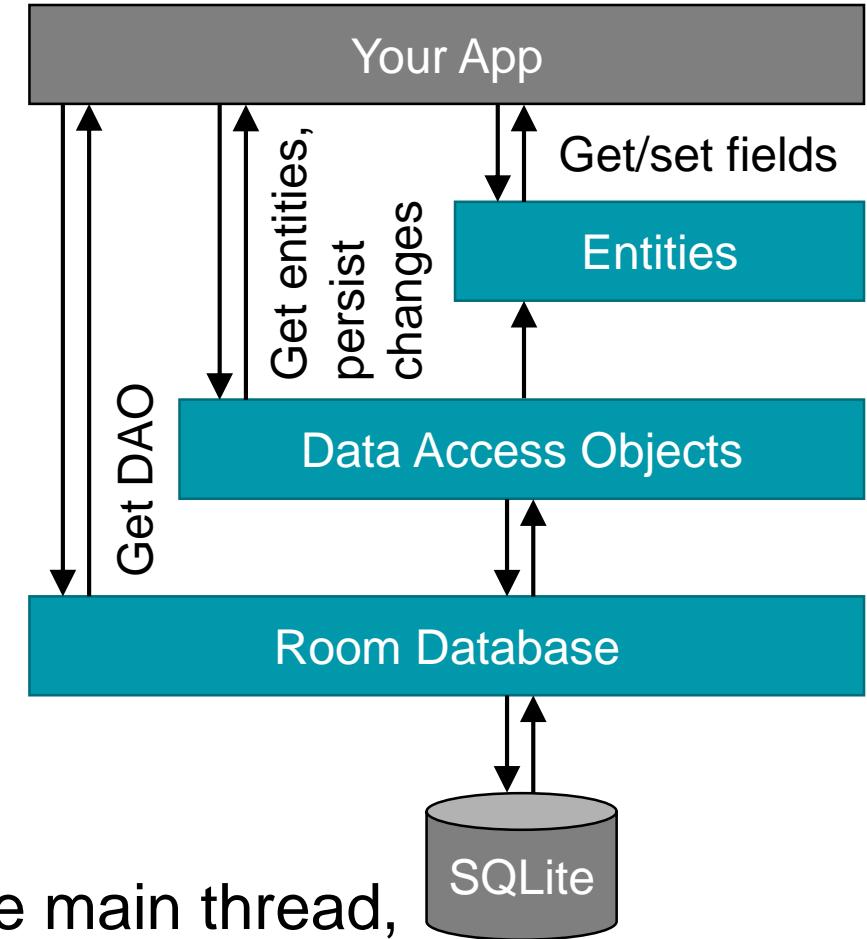
- Obtain an instance of the DAO:

```
UserDao userDao = db.userDao();
```

- Query database via DAO to receive entities:

```
List<User> users = userDao.getAll();
```

- Note: Room does not allow database access on the main thread, as it may delay user interface responses



Need to run queries in a background thread instead, as shown in previous lecture

Summary: Where to Store Your App's Data?

- Static media for integration in your app's user interface (e.g. images)
 - Resource directory of your project
- Volatile short-term storage of little data while activity is inactive (ie. activity state)
 - `savedInstanceState` bundle of an activity
- Persistent storage of simple data that only your app works with (e.g. game state)
 - `SharedPreferences` file in your app's private internal storage directory
- Persistent storage of large data that only your app works with (e.g. cached data)
 - Files in your app's private internal storage directory
- Dynamic, unstructured data that you exchange with other apps (e.g. documents)
 - Files in the user's external storage directories
- Dynamic, structured data that only your app works with (e.g. a to-do list)
 - Local SQLite database in your app's internal storage directory
- Data that you want to share with other users or devices (e.g. messages)
 - Central data storage on a remote server



In-Class Quiz 7: Where to Store Your App's Data?



- a) Static media for integration in your app's user interface (e.g. images)
- b) Volatile short-term storage of little data while activity is inactive (i.e. activity state)
- c) Persistent storage of simple data that only your app works with (e.g. game state)
- d) Persistent storage of large data that only your app works with (e.g. cached data)
- e) Dynamic, unstructured data that you exchange with other apps (e.g. documents)
- f) Dynamic, structured data that only your app works with (e.g. a to-do list)

Note the number of the most suitable storage mechanism:

1. Files in the user's external storage directories
2. Files in your app's private internal storage directory
3. Local SQLite database in your app's internal storage directory
4. Resource directory of your project
5. `savedInstanceState` bundle of an activity
6. `SharedPreferences` file in your app's private internal storage directory



Hugbúnaðarverkefni 2 / Software Project 2

10. Android User Interfaces: Fragments

HBV601G – Spring 2020

Matthias Book



HÁSKÓLI ÍSLANDS
VERKFRAÐI- OG NÁTTÚRVÍSINDASVIÐ
ÍÐNAÐARVERKFRAÐI-, VÉLAVERKFRAÐI-
OG TÖLVUNARFRÆÐIDEILD

Shift to Remote Teaching due to COVID-19

■ Lectures

- In classroom (incl. questions & quiz) – **phasing out**
- Live webcast (incl. questions & quiz) – **new addition**
 - Please use chat window in Panopto to ask questions remotely
- Recording (no questions & quiz) – **as usual**

■ Consultations

- ~~In classroom~~ – **discontinued**
- Teleconferencing – **as of today**
 - Please schedule individual appointments with tutors

■ Suggestions for improvement appreciated at book@hi.is



In-Class Quiz 8 Prep

- During class, I'll show you questions that you can answer with a number

| | | |
|-----------|----------|-------------|
| ID: _____ | @hi.is | Date: _____ |
| a) _____ | d) _____ | |
| b) _____ | e) _____ | |
| c) _____ | f) _____ | |



- **New:** Submit your quiz via Google Forms during lecture time.
- Submit answers on scrap of paper if you don't have network access.
- All questions in a quiz have same weight
- All quizzes (8-10 throughout semester) have the same weight
 - Your worst 2 quizzes will be disregarded
- Overall quiz grade counts as optional question worth 7.5% on final exam



Quiz 7 Solution: Where to Store Your App's Data?



- a) Static media for integration in your app's user interface (e.g. images)
 - b) Volatile short-term storage of little data while activity is inactive (i.e. activity state)
 - c) Persistent storage of simple data that only your app works with (e.g. game state)
 - d) Persistent storage of large data that only your app works with (e.g. cached data)
 - e) Dynamic, unstructured data that you exchange with other apps (e.g. documents)
 - f) Dynamic, structured data that only your app works with (e.g. a to-do list)
- (4) Resource directory of your project
 - (5) `savedInstanceState` bundle of an activity
 - (6) `SharedPreferences` file in your app's private internal storage directory
 - (2) Files in your app's private internal storage directory
 - (1) Files in the user's external storage directories
 - (3) Local SQLite database in your app's internal storage directory

Fragments

see also:

- Phillips et al.: Android Development, Ch. 7, 9 (2nd ed.) / 8 (3rd ed.), 10, 11, 17
- <http://developer.android.com/guide/components/fragments.html>



Preparation: Android Jetpack

- Different Android versions on different devices offer different “built-in” features
- By incorporating Android Jetpack components, an app can “bring along” API features that may not be built into older Android versions
 - Increase backward compatibility
 - Rely on features only present in the Jetpack library
 - Benefit from faster updates of Jetpack library than updates rolled out to devices
- By using `androidx.fragment.app.Fragment` and `.FragmentActivity` instead of the built-in `android.app.Fragment` and `.FragmentActivity`, we make sure our app runs on more devices and benefits from updates earlier.



Preparation: Adding Architecture Components

- Add Google's Maven repository to your *project's build.gradle* file:
 - i.e. e.g. for CrimeLab, to **CrimeLab\build.gradle**

```
allprojects {  
    repositories {  
        google()  
        jcenter()  
    }  
}
```

- Add the Fragment library dependency to your *app's build.gradle* file:
 - i.e. e.g. for CrimeLab, to **CrimeLab\app\build.gradle**

```
dependencies {  
    implementation "androidx.fragment:fragment:1.2.2"  
}
```

See
<https://developer.android.com/jetpack/docs/getting-started>
for details



Legacy Solution: Android Support Library

- Before Android Jetpack, the Android Support Library fulfilled the same purpose.
- The **-vX** suffix of a support library indicates the API level it requires
 - e.g. `support-v4` can be used on devices running API level 4 (i.e. Android 1.6) or higher
- Using `android.support.v4.app.Fragment` and `.FragmentActivity` instead of the built-in `android.app.Fragment` and `.FragmentActivity` made sure apps run on more devices and benefit from updates earlier.
- Note the Android textbook examples still use the support library!
 - See <https://developer.android.com/topic/libraries/support-library/packages#v4-fragment> for a description of the support library
 - See <https://developer.android.com/jetpack/androidx/migrate> on how to migrate from the support library to Jetpack components

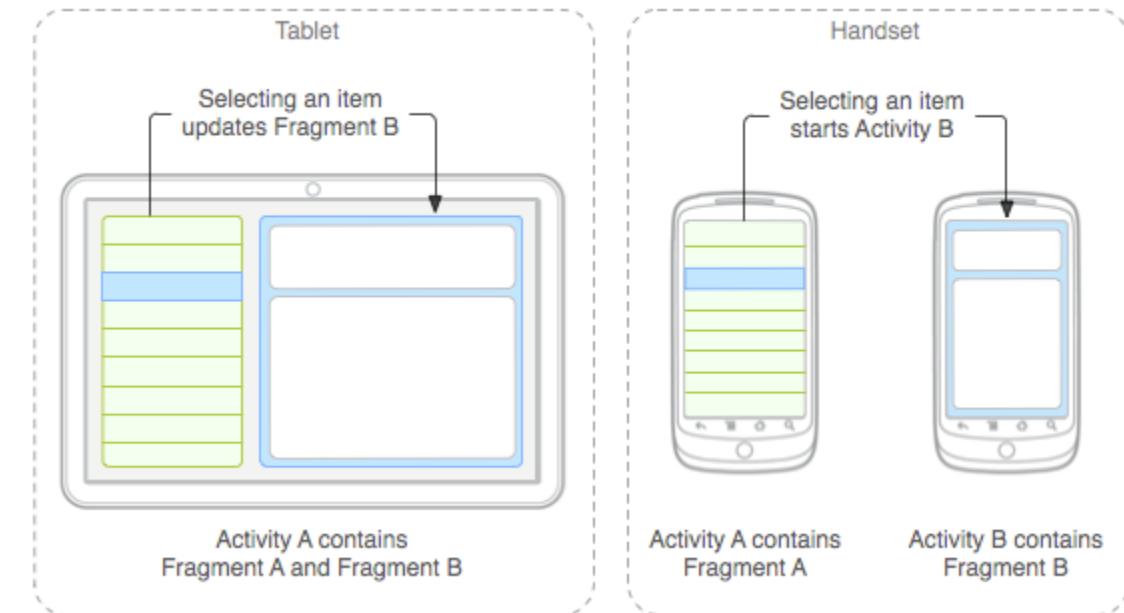
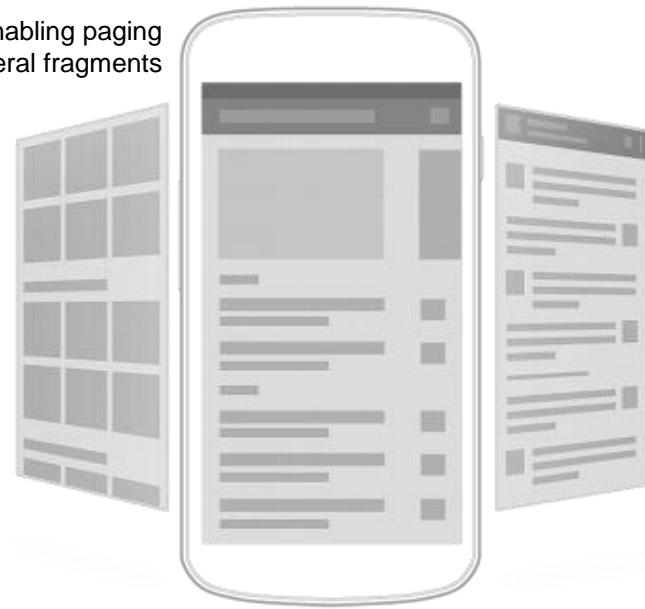


Activities vs. Fragments

- An **activity** is the controller for one particular screen with a static layout
- A **fragment** is the controller for a quite independent user interface “module” that
 - is nested into (“hosted by”) a host activity
 - can be added to or deleted from the host activity at runtime
 - has its own layout and behavior
 - receives its own input and lifecycle events
 - can be shown in combination with other fragments inside one activity
 - can be shown as a pop-up or tabbed dialog

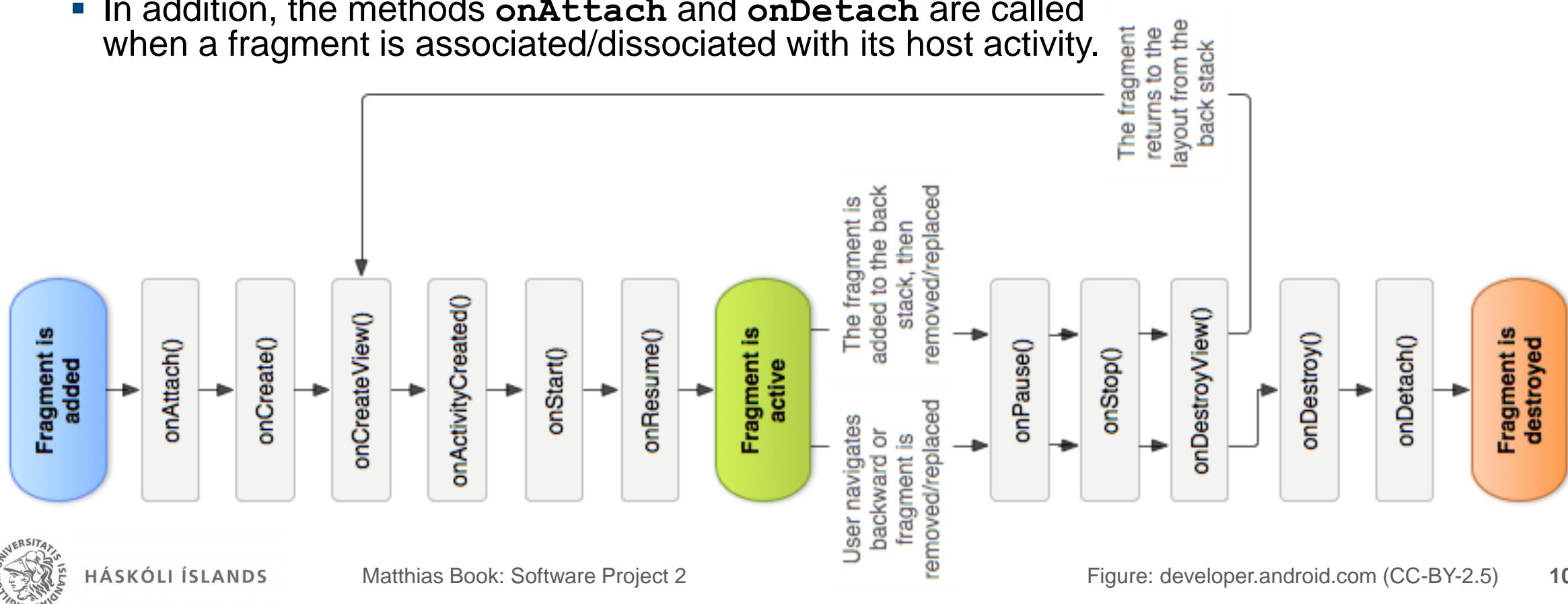
➤ Fragments enable dynamic UI changes.

One activity enabling paging through several fragments



Fragment Lifecycle

- A fragment's lifecycle methods (**onCreate**, **onPause** etc.) mirror an activity's.
- When an activity's lifecycle method is called, the same method is also called on the fragment(s) hosted by the activity.
 - In addition, the methods **onAttach** and **onDetach** are called when a fragment is associated/dissociated with its host activity.



In Principle, It's Easy: Creating a Fragment

- To create a fragment, extend the `Fragment` class and inflate the fragment's layout in its `onCreateView` method:

```
import android.os.Bundle;
import androidx.fragment.app.Fragment;
import android.view.LayoutInflater;
import android.view.ViewGroup;

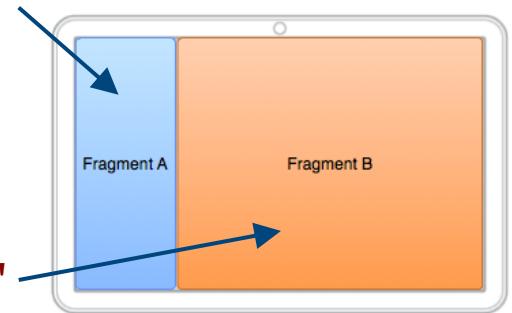
public class ArticleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
        return inflater.inflate(R.layout.article_view, container, false);
    }
}
```



- The layout for a fragment is constructed in the same way as the layout for an activity (*not shown here; see Lecture 5*)

The Easy Way: Including Fragments Statically in Layout

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="horizontal"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent">  
  
    <fragment android:name="com.example.android.fragments.HeadlinesFragment"  
        android:id="@+id/headlines_fragment"  
        android:layout_weight="1"  
        android:layout_width="0dp"  
        android:layout_height="match_parent" />  
  
    <fragment android:name="com.example.android.fragments.ArticleFragment"  
        android:id="@+id/article_fragment"  
        android:layout_weight="2"  
        android:layout_width="0dp"  
        android:layout_height="match_parent" />  
  
</LinearLayout>
```



Note: Fragments added to a layout this way cannot be removed or swapped out at runtime!

The Easy Way: Main Activity Controlling Overall Layout

- The layout we just defined still needs an associated activity:

```
import android.os.Bundle;
import androidx.fragment.app.FragmentActivity;

public class MainActivity extends FragmentActivity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.news_articles);
    }
}
```

Note: Need to inherit from **FragmentActivity** if the layout contains fragments.

Inflate the layout we just defined.

- Note: The activity controls the lifecycle and events of the overall layout, but cannot influence fragments that are defined directly in the layout XML.
 - To have control of the fragments at runtime, we need to add them at runtime.
 - This is considerably more complex, but also more flexible.

In Practice, It's More Complex...

Example: Building a List-Detail Interface with Fragments

- **User interface components:**

- One “list” fragment showing a list of several items
- One “detail” fragment showing the details of one item

- **Intended behavior:**

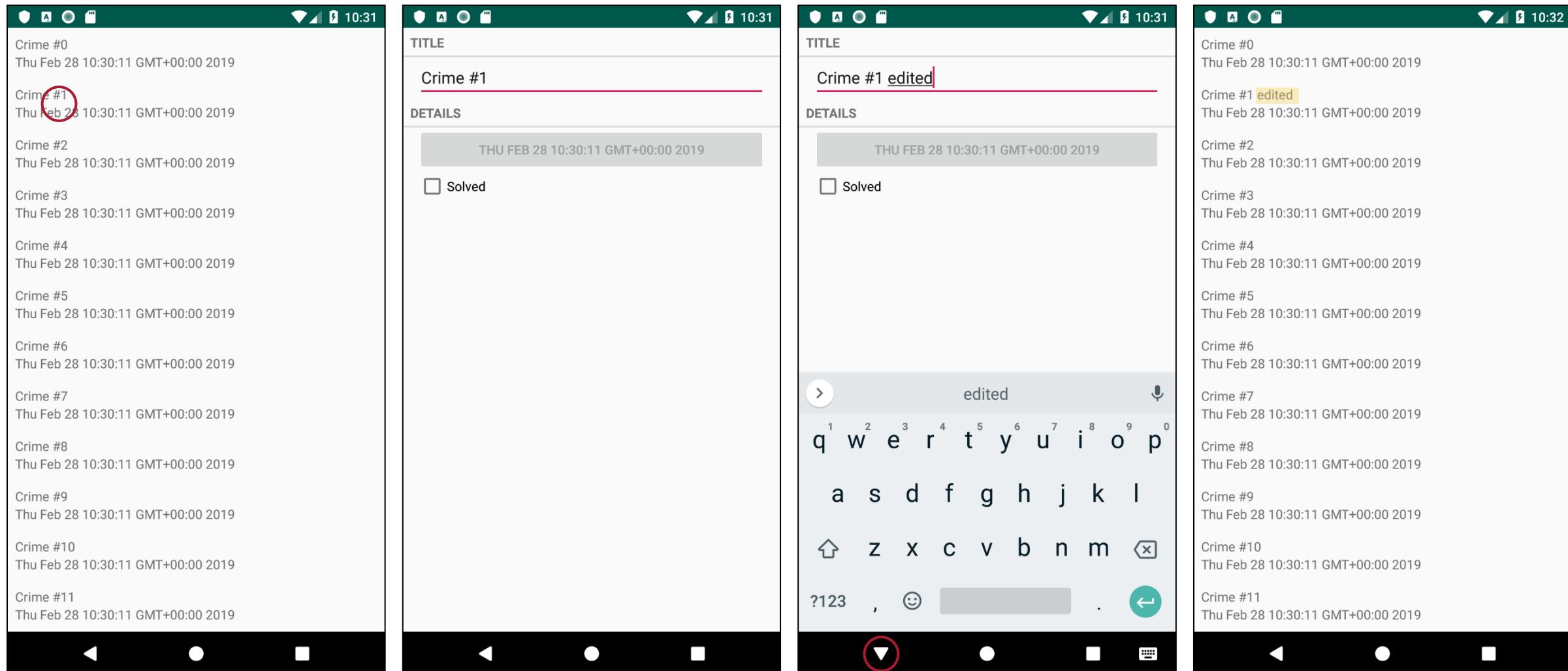
- Selecting one item in the “list” fragment should show its details in the “detail” fragment
- Editing information in the “detail” fragment should be reflected in the “list” fragment

- **Layout flexibility:**

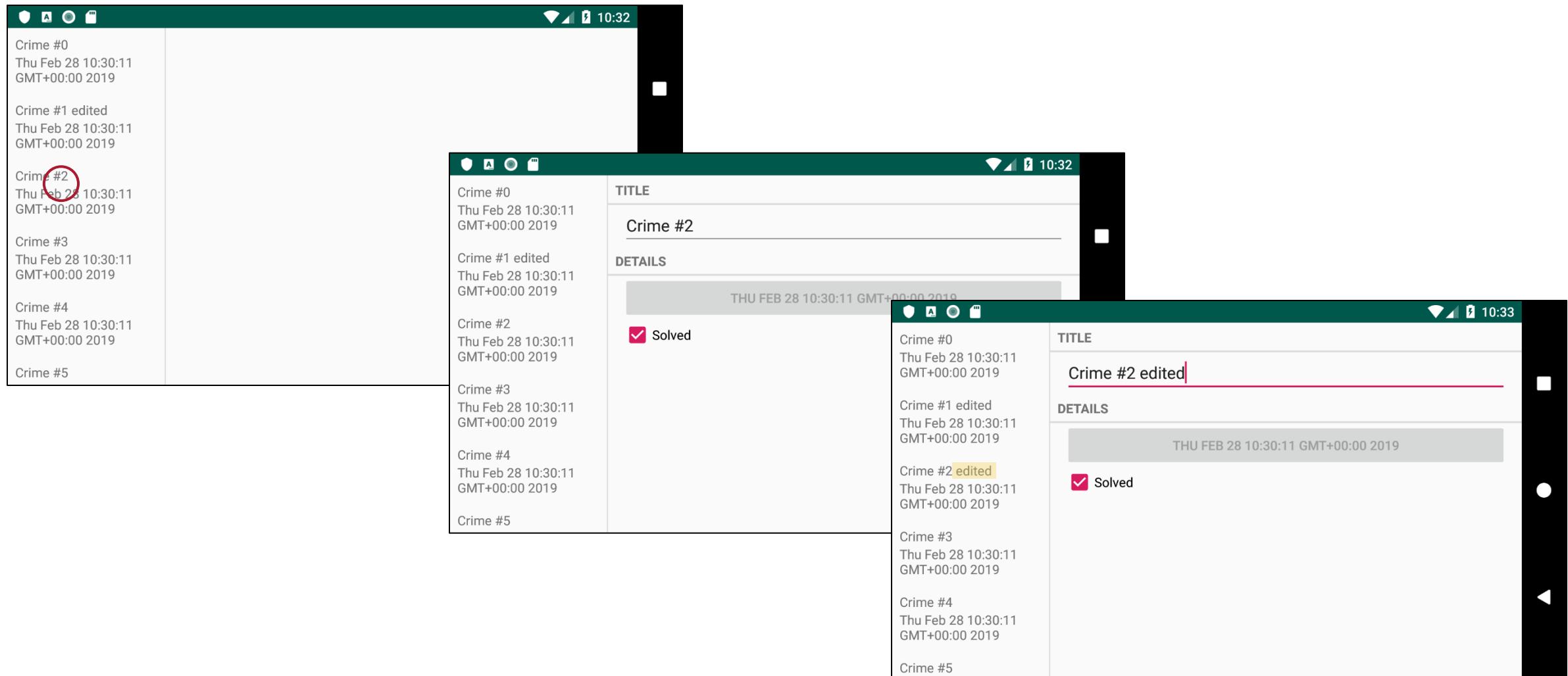
- In portrait orientation, only one of the fragments should be visible at any time
- In landscape orientation, both fragments should be visible side by side



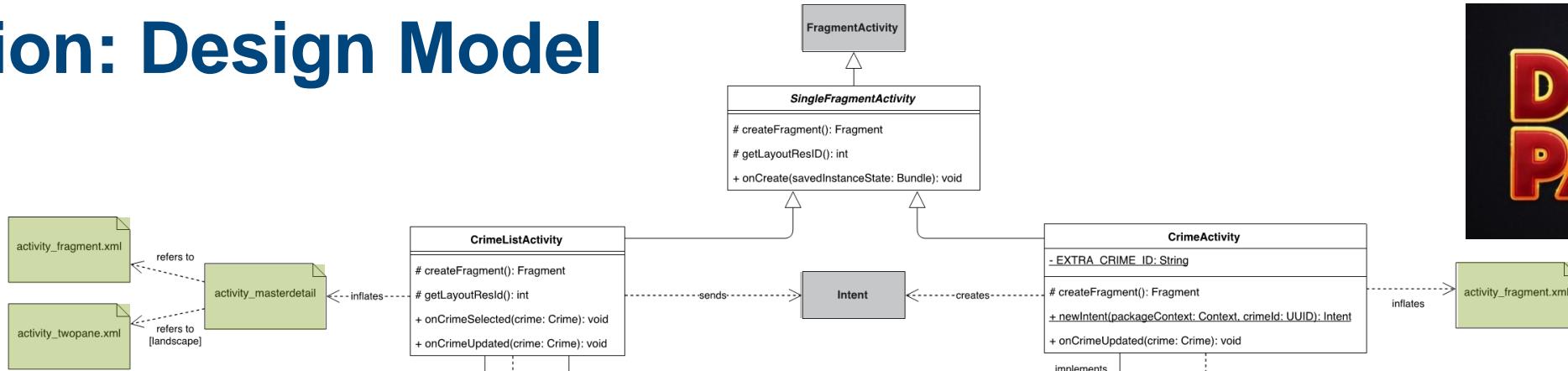
Requirement: Layout and Behavior in Portrait Orientation



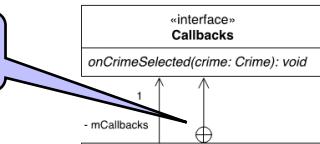
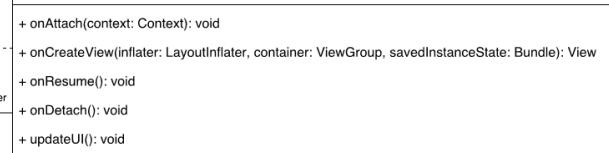
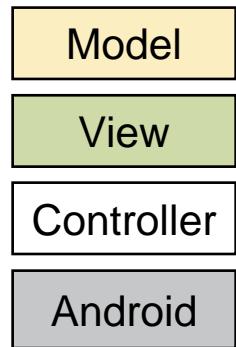
Requirement: Layout and Behavior in Landscape Orient.



Solution: Design Model



Inner class/interface

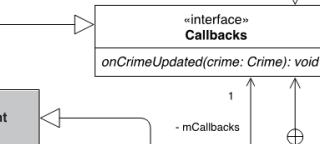


implements

implements

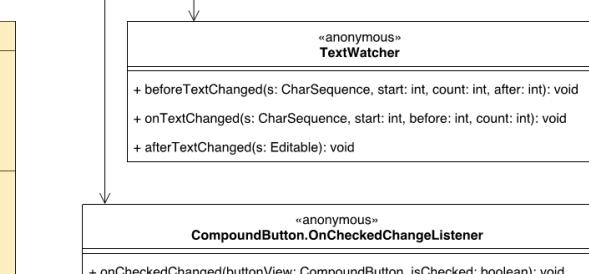
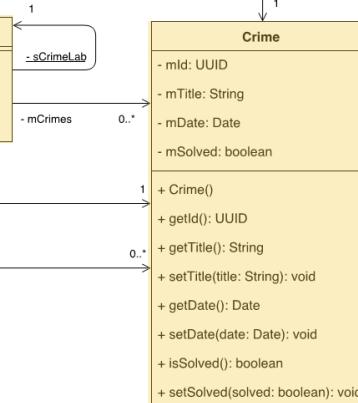
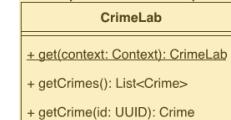
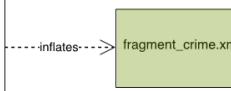
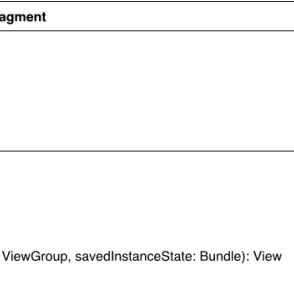
hosts

hosts



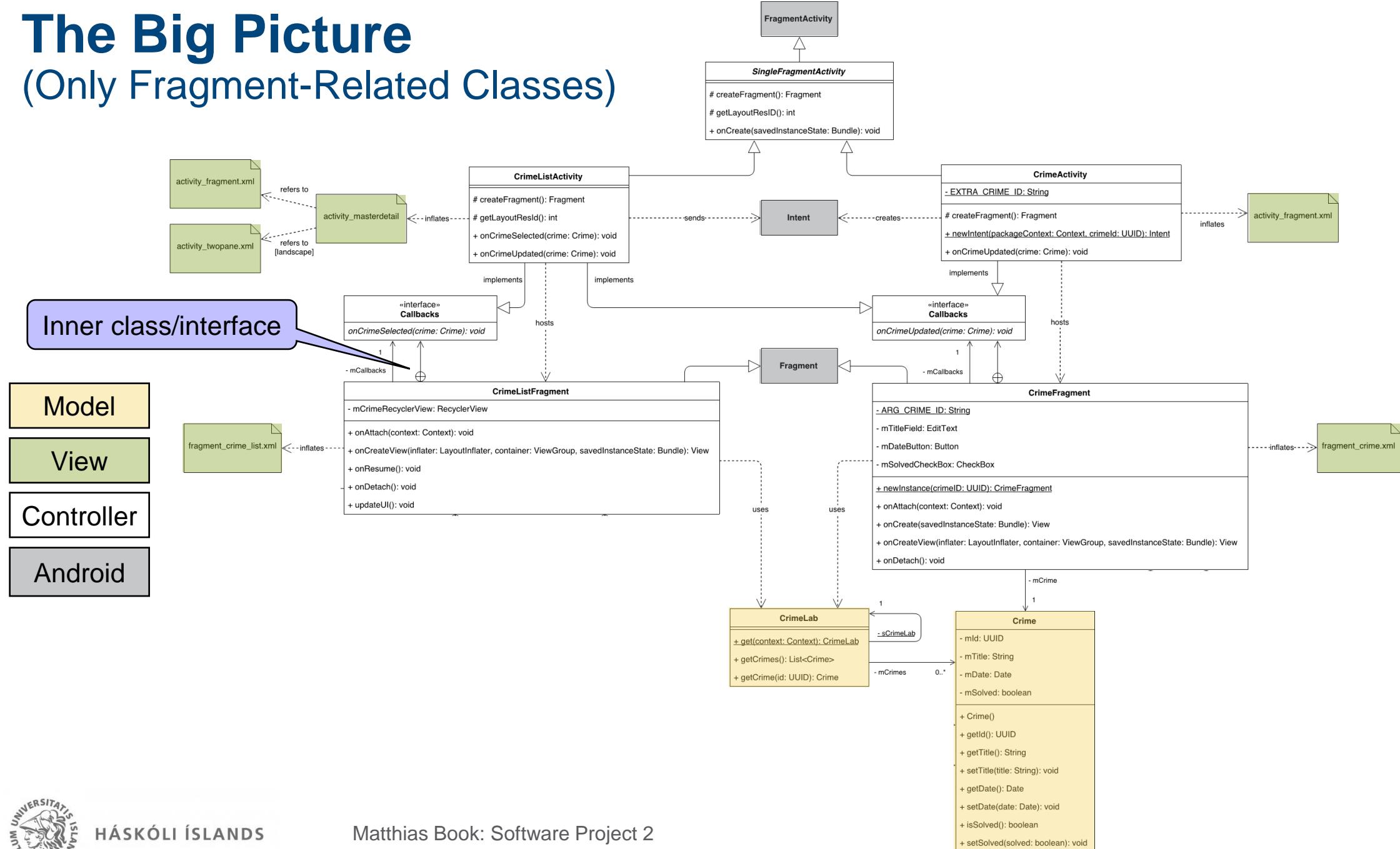
hosts

hosts

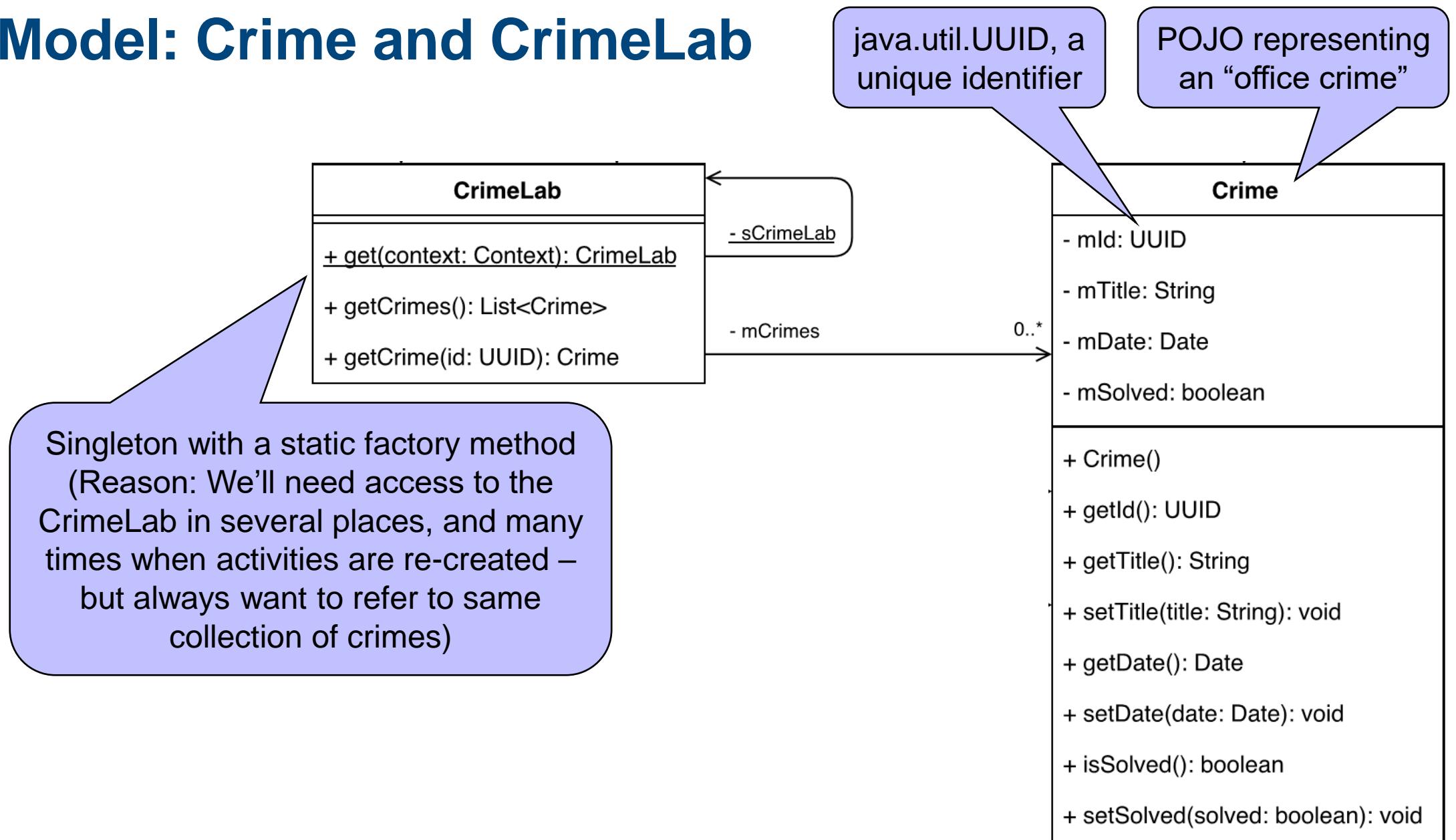


The Big Picture

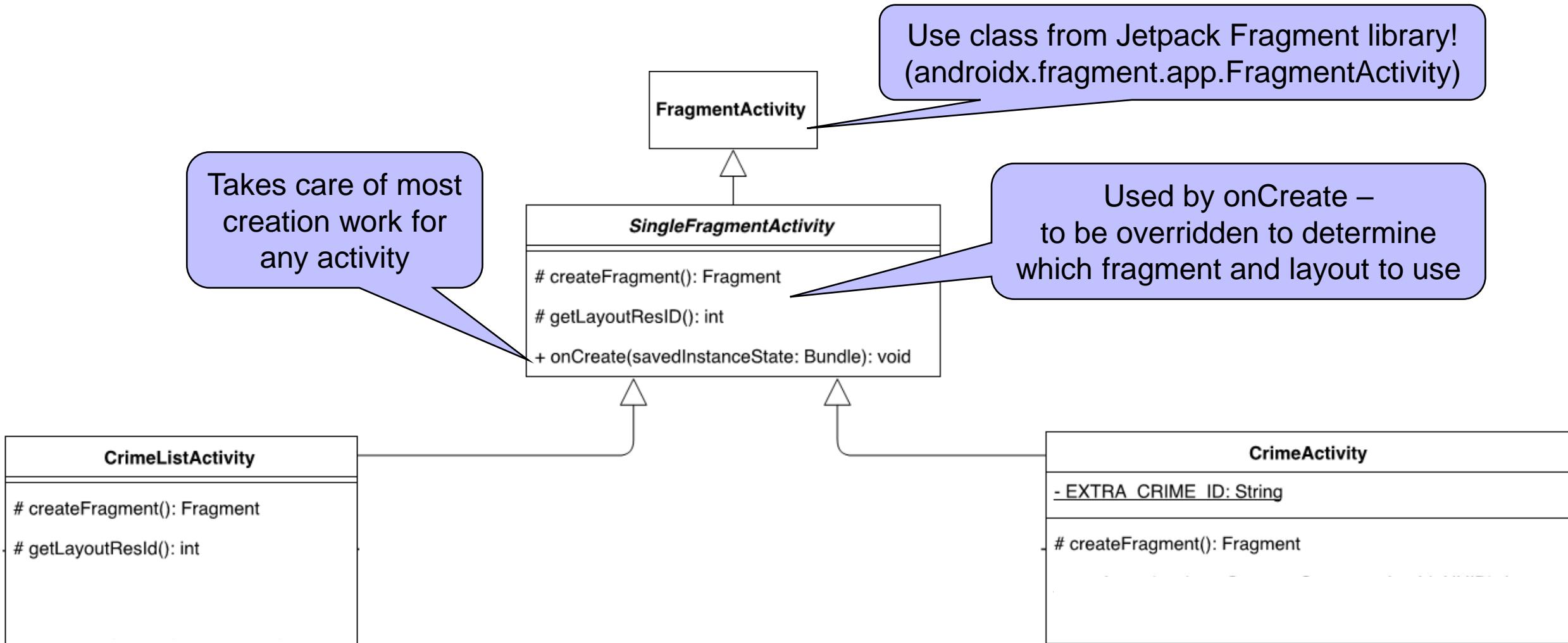
(Only Fragment-Related Classes)



The Model: Crime and CrimeLab



The Activities: CrimeListActivity and CrimeActivity



The Abstract Superclass SingleFragmentActivity

```
public abstract class SingleFragmentActivity extends FragmentActivity {  
  
    Used by onCreate –  
    to be overridden to  
    determine which fragment  
    and layout to use  
  
    protected abstract Fragment createFragment();  
  
    @LayoutRes  
    protected int getLayoutResId() {  
        return R.layout.activity_fragment;  
    }  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(getLayoutResId());  
  
        FragmentManager fm = getSupportFragmentManager();  
        Fragment fragment = fm.findFragmentById(R.id.fragment_container);  
  
        if (fragment == null) {  
            fragment = createFragment();  
            fm.beginTransaction()  
                .add(R.id.fragment_container, fragment)  
                .commit();  
        }  
    }  
}
```

This can be used in
any project to
incorporate single
fragments into their
host activities

Set the layout

Try to retrieve the fragment
from the layout

Instantiate fragment and
insert it into the layout

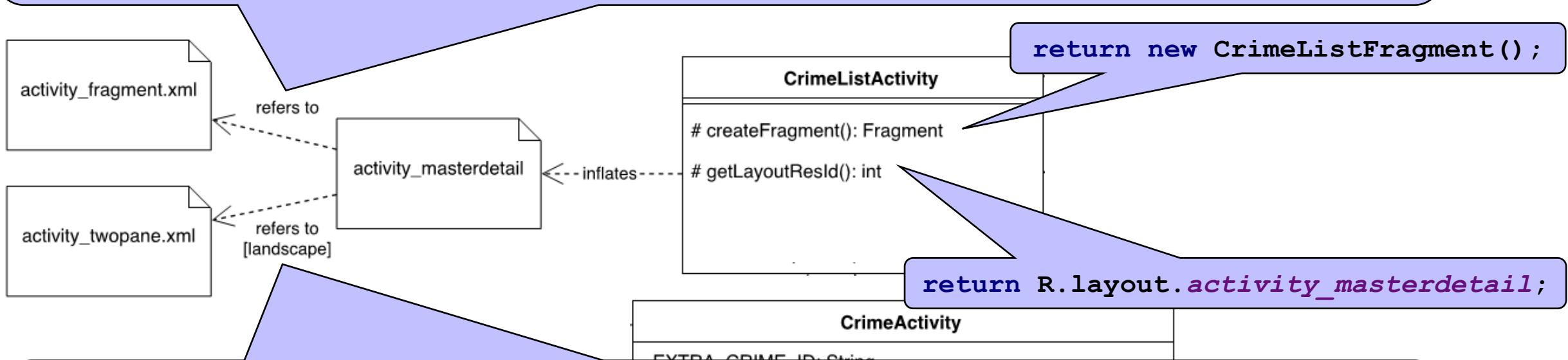
Fragment may already exist
if activity is re-created



The Activity Layouts: activity_fragment and activity_twopane

res/values/refs.xml (default: portrait):

```
<resources>
    <item name="activity_masterdetail" type="layout">@layout/activity_fragment</item>
</resources>
```



res/values-land/refs.xml (landscape):

```
<resources>
    <item name="activity_masterdetail" type="layout">@layout/activity_twopane</item>
</resources>
```

The Activity Layouts: `activity_fragment` and `activity_twopane`

`activity_fragment.xml`

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/fragment_container"  
    android:layout_width="match_parent" android:layout_height="match_parent"/>
```

Generic one-pane layout

`activity_twopane.xml`

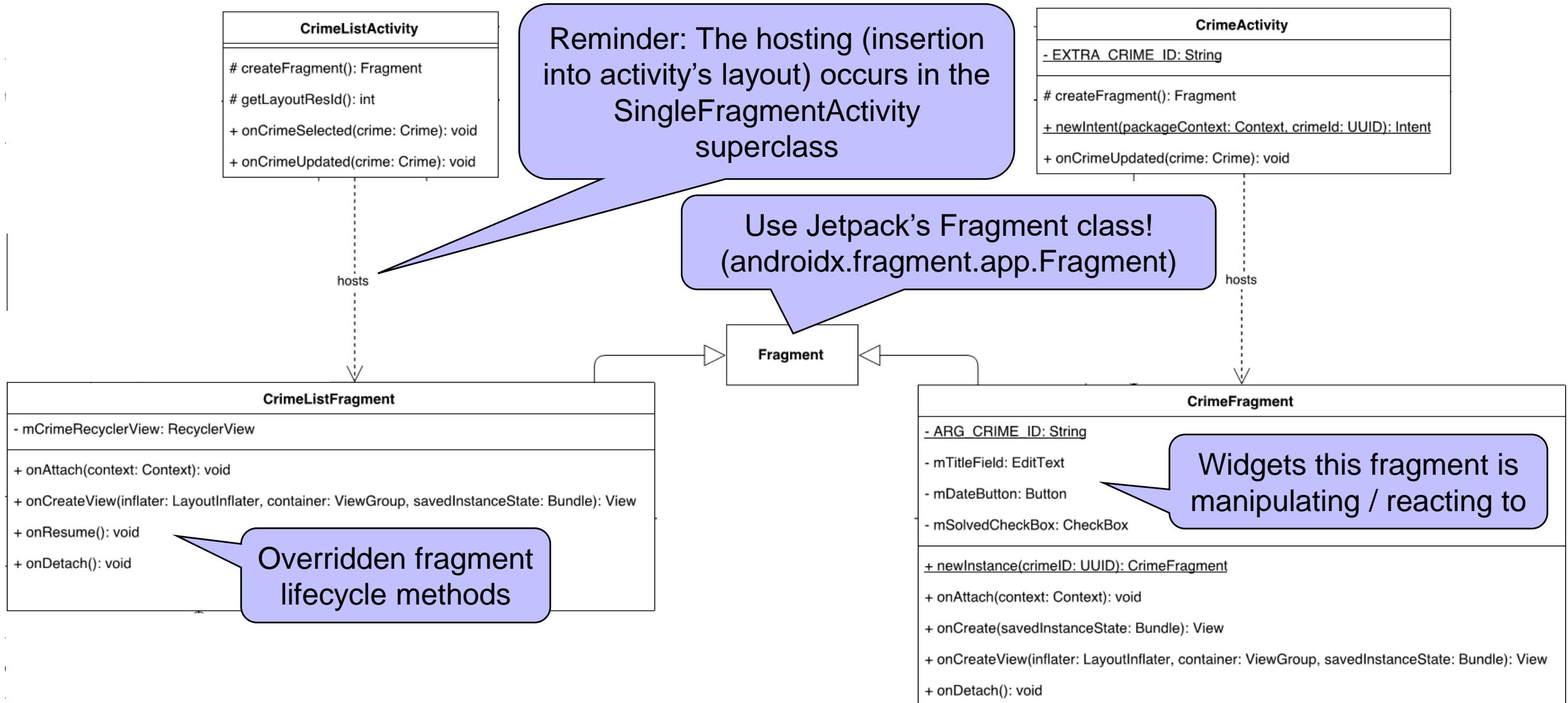
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent" android:layout_height="match_parent"  
    android:divider="?android:attr/dividerHorizontal" android:showDividers="middle"  
    android:orientation="horizontal">  
    <FrameLayout android:id="@+id/fragment_container"  
        android:layout_width="0dp" android:layout_height="match_parent"  
        android:layout_weight="1"/>  
    <FrameLayout android:id="@+id/detail_fragment_container"  
        android:layout_width="0dp" android:layout_height="match_parent"  
        android:layout_weight="3"/>  
</LinearLayout>
```

Two panes with 1:3 size relation

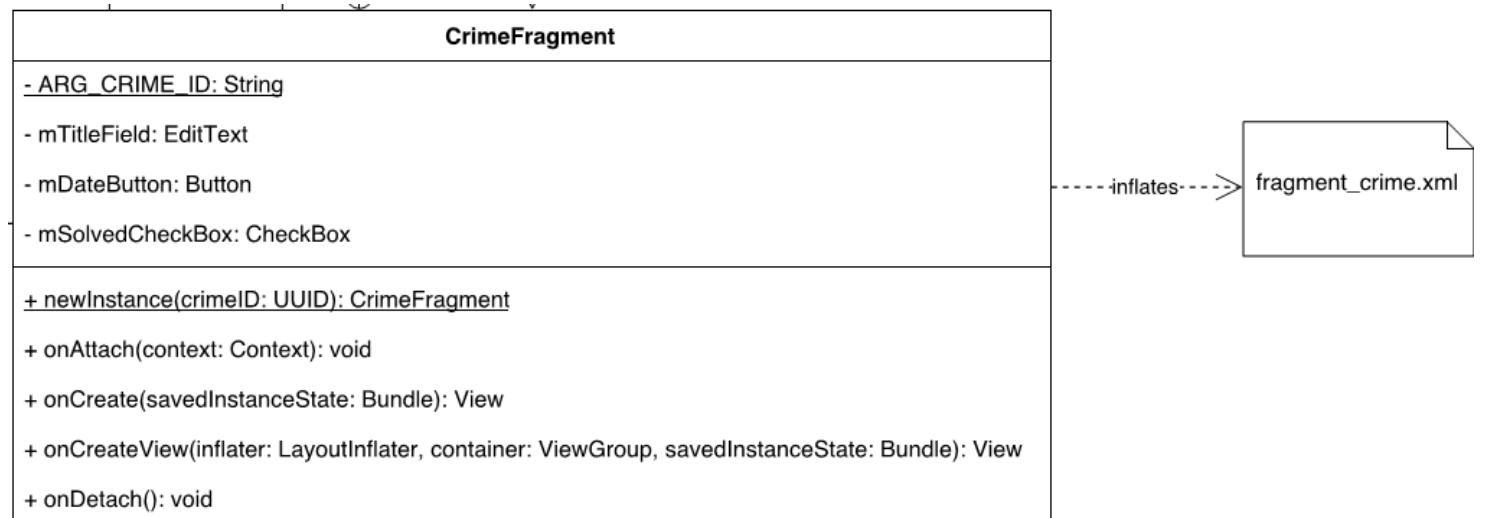
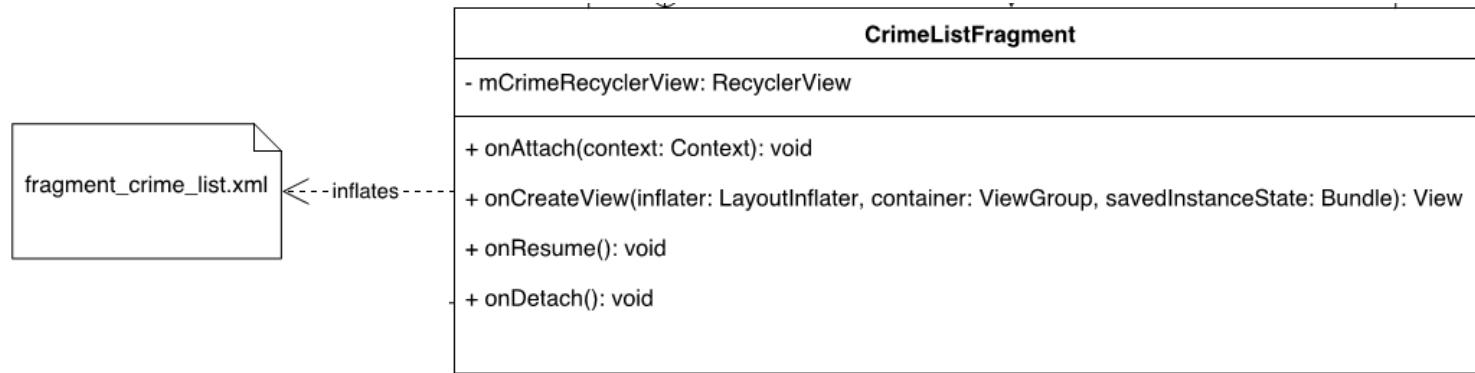
Note: `SingleFragmentActivity` will only populate the `fragment_container`. The `detail_fragment_container` initially remains empty.



The Fragments: CrimeListFragment and CrimeFragment



The Fragment Layouts: fragment_crime_list and fragment_crime



The Fragment Layouts: fragment_crime_list and fragment_crime

Layout of the list *items* will be defined in separate file
(see Android textbook, Ch. 8)

List layout

fragment_crime_list.xml

```
<android.support.v7.widget.RecyclerView xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/crime_recycler_view"  
    android:layout_width="match_parent" android:layout_height="match_parent"/>
```

Detail layout

fragment_crime.xml

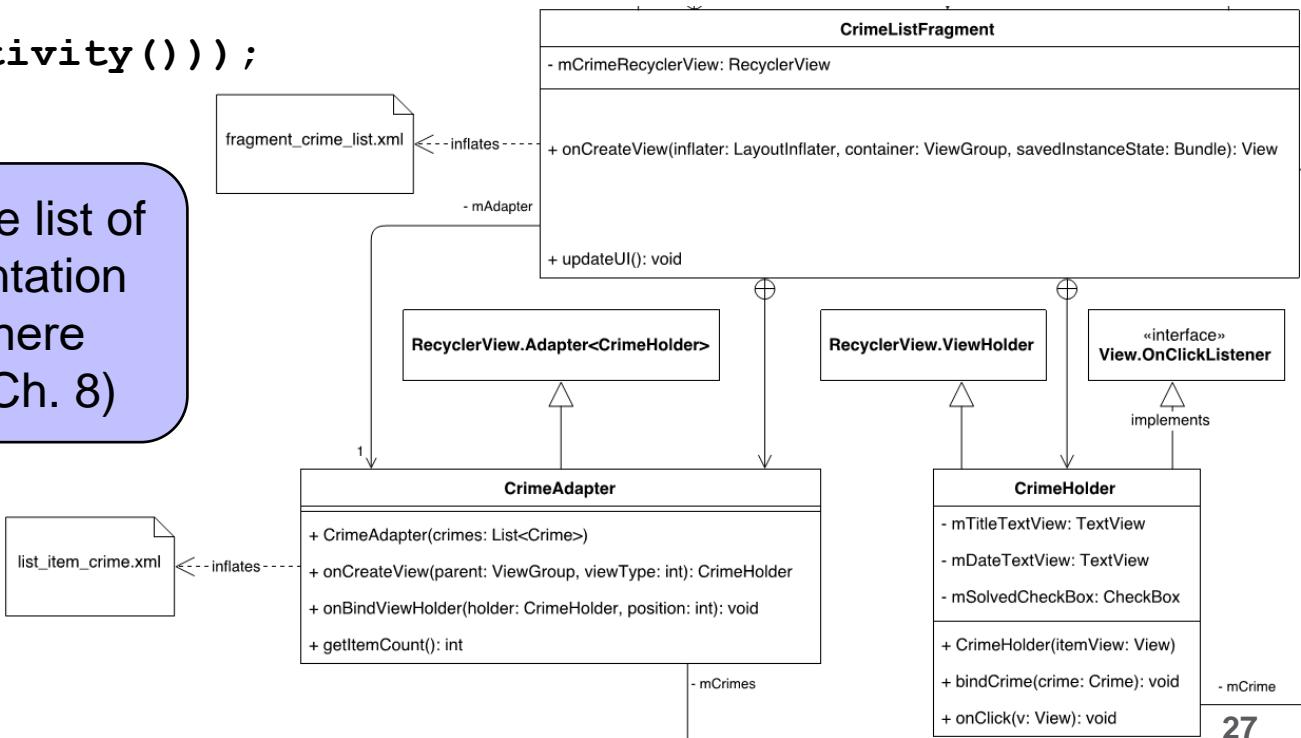
```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:orientation="vertical"  
    android:layout_width="match_parent" android:layout_height="match_parent">  
  
    <TextView android:text="@string/crime_title_label"  
        android:layout_width="match_parent" android:layout_height="wrap_content"  
        style="?android:listSeparatorTextViewStyle"/>  
  
    <EditText android:id="@+id/crime_title"  
        android:layout_width="match_parent" android:layout_height="wrap_content"  
        android:layout_marginLeft="16dp" android:layout_marginRight="16dp"  
        android:hint="@string/crime_title_hint"/>  
  
    <TextView android:text="@string/crime_details_label"  
        android:layout_width="match_parent" android:layout_height="wrap_content"  
        style="?android:listSeparatorTextViewStyle"/>  
  
    <Button android:id="@+id/crime_date"  
        android:layout_width="match_parent" android:layout_height="wrap_content"  
        android:layout_marginLeft="16dp" android:layout_marginRight="16dp"/>  
  
    <CheckBox android:id="@+id/crime_solved"  
        android:layout_width="match_parent" android:layout_height="wrap_content"  
        android:layout_marginLeft="16dp" android:layout_marginRight="16dp"  
        android:text="@string/crime_solved_label"/>  
  
    </LinearLayout>
```



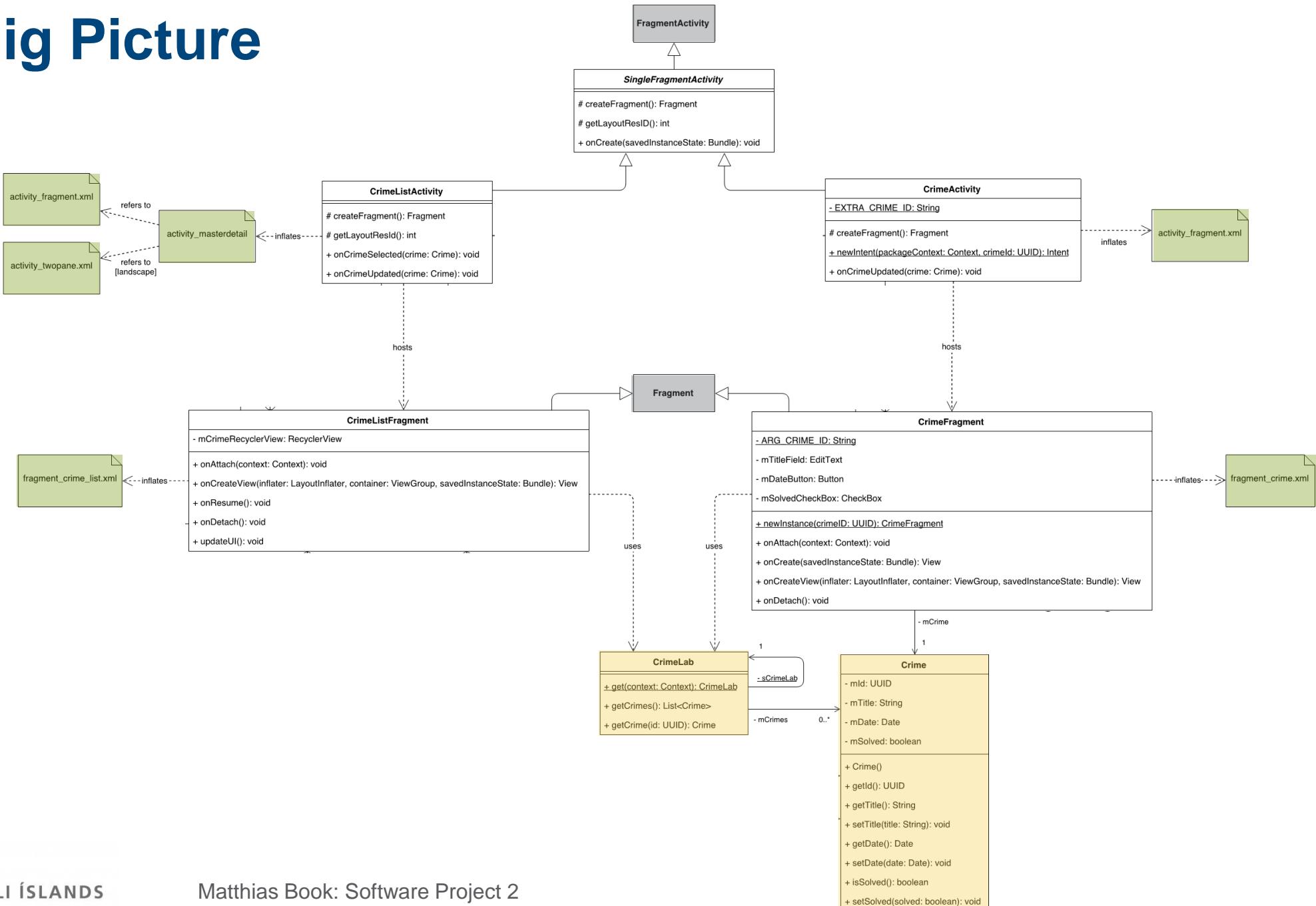
Creating the List Fragment: CrimeListFragment.onCreateView

```
@Override  
public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                         Bundle savedInstanceState) {  
    View view = inflater.inflate(R.layout.fragment_crime_list, container, false);  
  
    mCrimeRecyclerView = (RecyclerView) view  
        .findViewById(R.id.crime_recycler_view);  
    mCrimeRecyclerView.setLayoutManager(  
        new LinearLayoutManager(getActivity()));  
  
    updateUI();  
  
    return view;  
}
```

This creates the scrollable list of crimes, whose implementation details we'll skip over here (see Android textbook, Ch. 8)

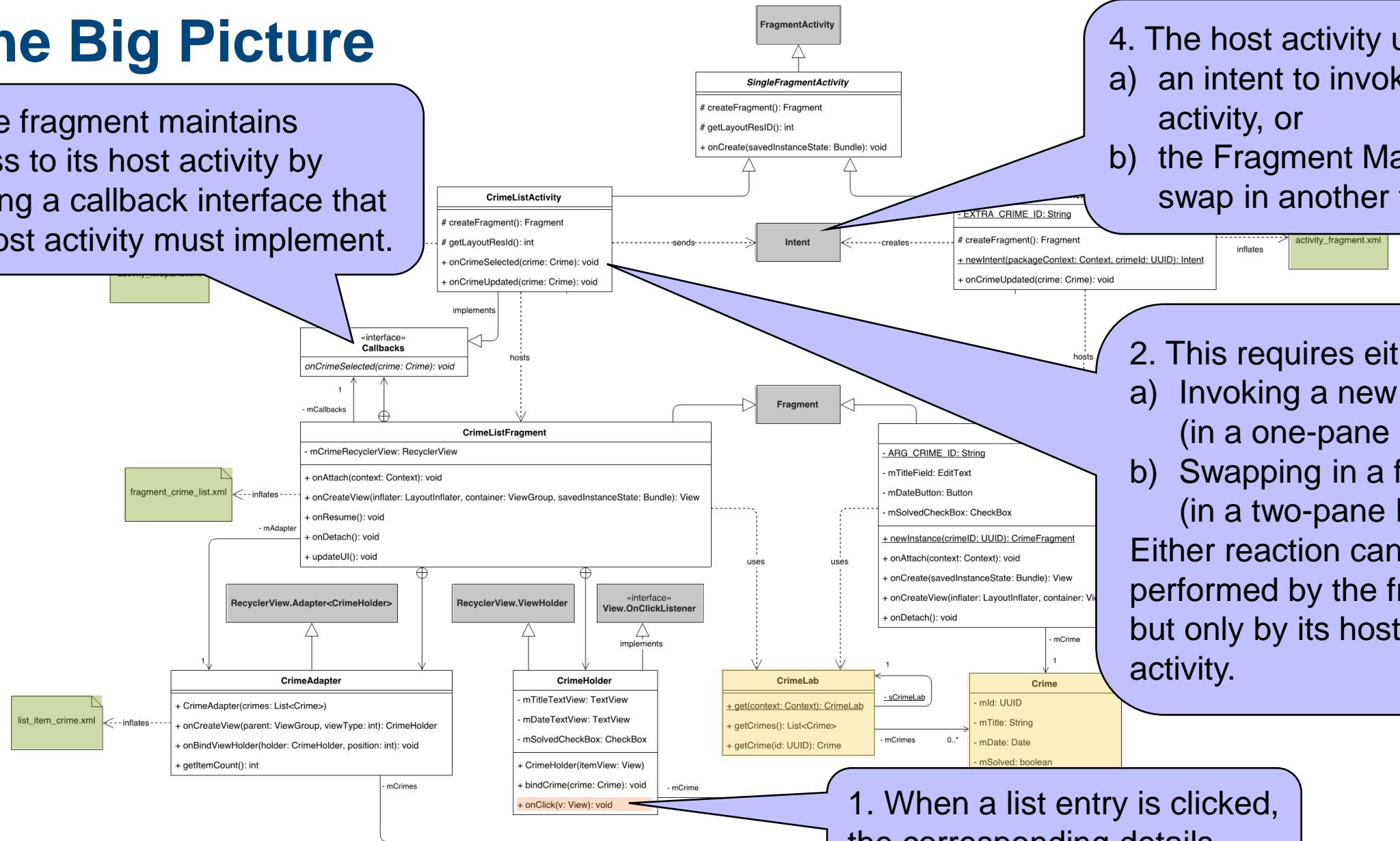


The Big Picture



The Big Picture

3. The fragment maintains access to its host activity by defining a callback interface that the host activity must implement.



4. The host activity uses either:

- an intent to invoke another activity, or
- the Fragment Manager to swap in another fragment

2. This requires either:

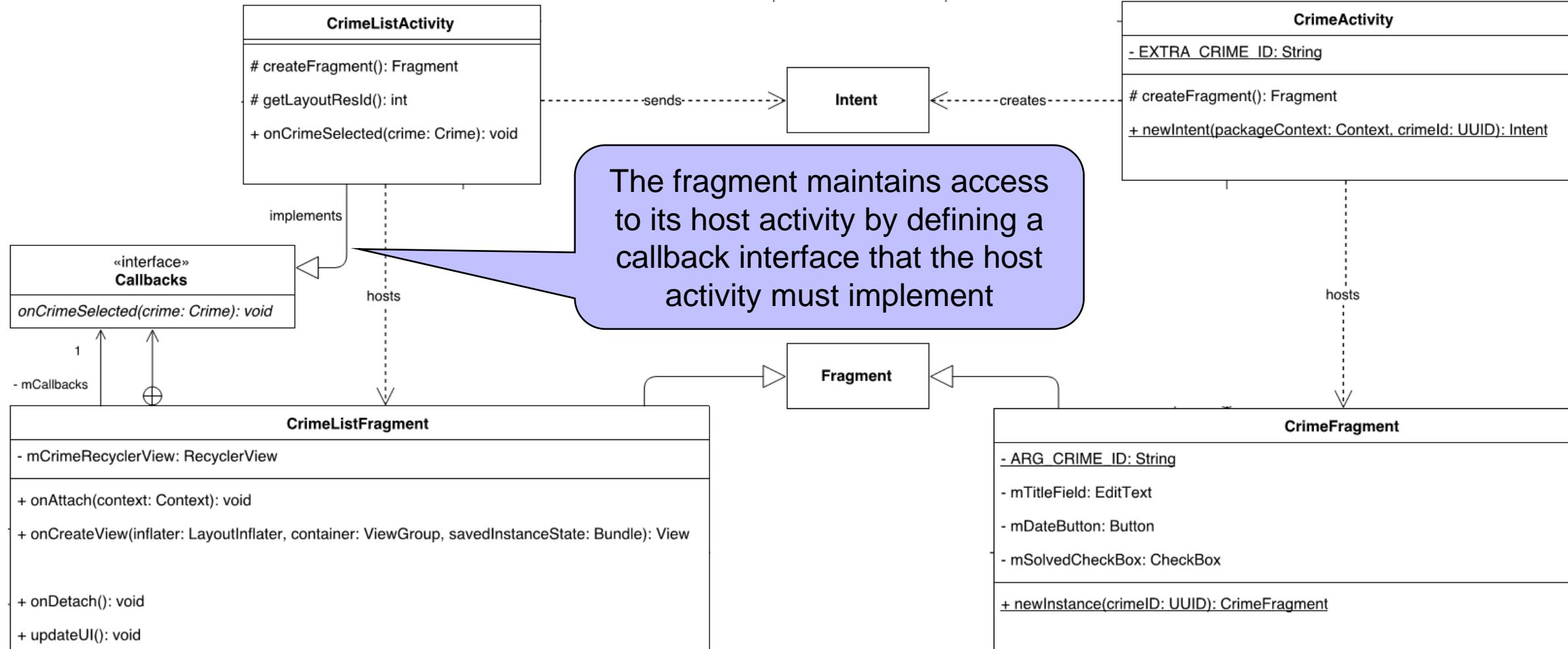
- Invoking a new activity (in a one-pane layout), or
- Swapping in a fragment (in a two-pane layout)

Either reaction cannot be performed by the fragment, but only by its hosting activity.

1. When a list entry is clicked, the corresponding details shall be displayed.

The List's Callback Interface: CrimeListFragment.Callbacks

This is how a fragment sends messages to its host activity



Declaring the List's Callback Interface: CrimeListFragment.Callbacks

```
public class CrimeListFragment extends Fragment {  
  
    private Callbacks mCallbacks;  
  
    public interface Callbacks {  
        void onCrimeSelected(Crime crime);  
    }  
  
    @Override  
    public void onAttach(Context context) {  
        super.onAttach(context);  
        mCallbacks = (Callbacks) context;  
    }  
  
    @Override  
    public void onDetach() {  
        super.onDetach();  
        mCallbacks = null;  
    }  
  
    // ...  
}
```

Reference to host activity

Declaration of interface that the host activity must implement to react to a selection of a list entry

Automatically stores a reference to the host activity (here: CrimeListActivity) when the fragment is associated with it

Automatically removes the reference to the host activity when the fragment is dissociated from it



Invoking the List's Callback Interface: CrimeListFragment.CrimeHolder.onClick

See Android textbook, Ch. 8
for scrolling list implementation
details

The scrolling list's event handler
for clicks on an item

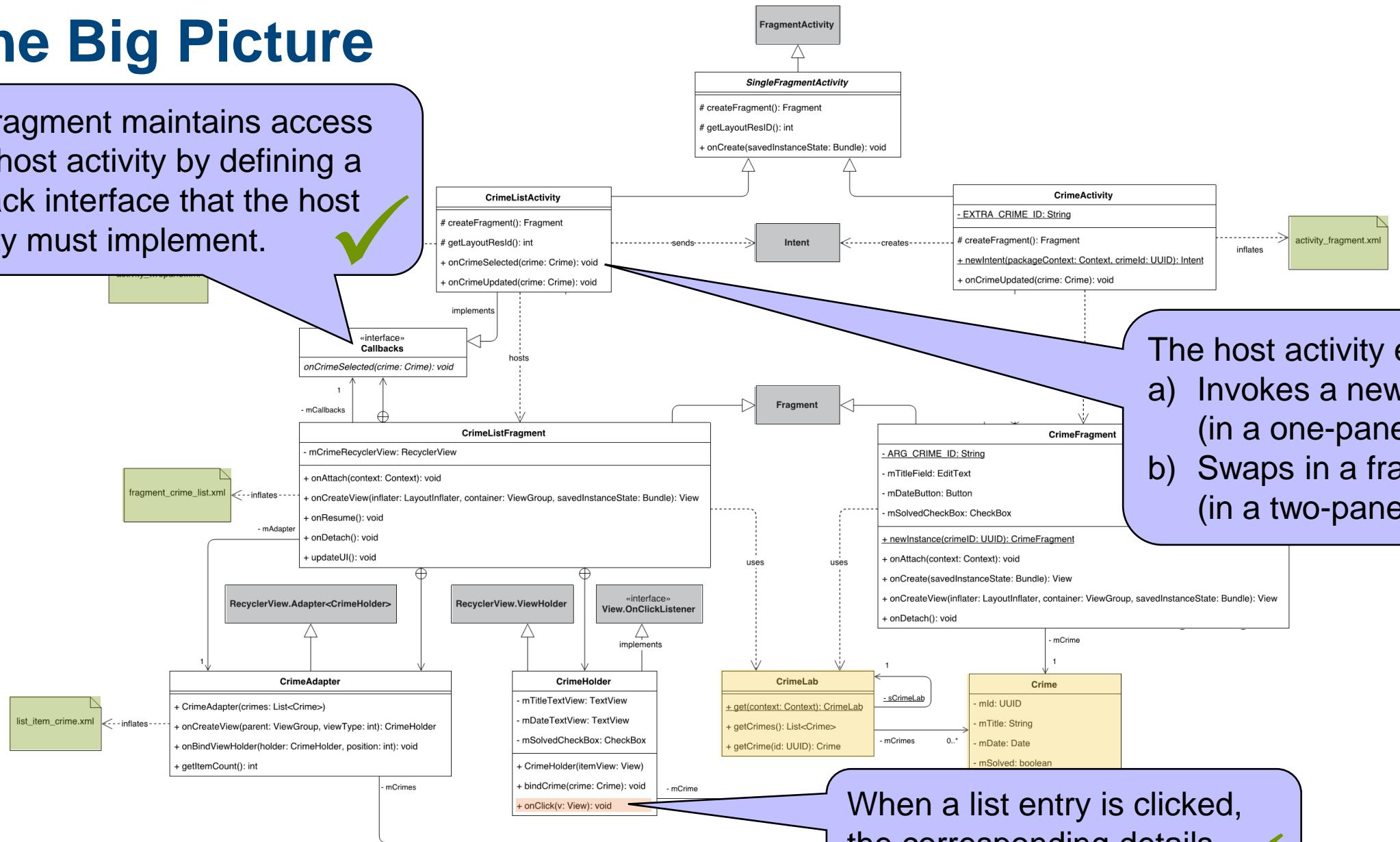
Notify host activity (via callback
reference) of selection of an item

```
private class CrimeHolder  
    extends RecyclerView.ViewHolder  
    implements View.OnClickListener {  
  
    private Crime mCrime;  
  
    @Override  
    public void onClick(View v) {  
        mCallbacks.onCrimeSelected(mCrime);  
    }  
    // ...  
}
```



The Big Picture

The fragment maintains access to its host activity by defining a callback interface that the host activity must implement.



The host activity either:

- Invokes a new activity (in a one-pane layout), or
- Swaps in a fragment (in a two-pane layout)

When a list entry is clicked, the corresponding details shall be displayed.

Implementing the List's Callback Interface: CrimeListActivity.onCrimeSelected

```
public class CrimeListActivity extends SingleFragmentActivity  
    implements CrimeListFragment.Callbacks {  
  
    @Override  
    public void onCrimeSelected(Crime crime) {  
        if (findViewById(R.id.detail_fragment_container) == null) {  
            Intent intent = CrimeActivity.newIntent(this, crime.getId());  
            startActivity(intent);  
        } else {  
            Fragment newDetail = CrimeFragment.newInstance(crime.getId());  
            getSupportFragmentManager().beginTransaction()  
                .replace(R.id.detail_fragment_container, newDetail)  
                .commit();  
        }  
    }  
}  
// ...
```

This is how an activity
invokes another
activity or fragment

Check if the fragment container
for the detail view exists
(this is only the case if the two-
pane layout was inflated earlier)

One-pane interface:
Use intent to start new
activity and pass ID of
selected item along

Two-pane interface:
Use fragment manager to replace
fragment in detail container, and
pass ID of selected item along



Providing the Intent and Reacting to It (To Show Details in One-Pane Layout)

This is how data is passed to and received by an activity: through intent extras

```
public class CrimeActivity extends SingleFragmentActivity {  
  
    private static final String EXTRA_CRIME_ID =  
        "com.bignerdranch.android.criminalintent.crime_id";  
  
    public static Intent newIntent(Context packageContext, UUID crimeId) {  
        Intent intent = new Intent(packageContext, CrimeActivity.class);  
        intent.putExtra(EXTRA_CRIME_ID, crimeId);  
        return intent;  
    }  
  
    @Override  
    protected Fragment createFragment() {  
        UUID crimeId = (UUID) getIntent().getSerializableExtra(EXTRA_CRIME_ID);  
        return CrimeFragment.newInstance(crimeId);  
    }  
}
```

Typo-prevention constant

Pass desired item ID along as parameter

Static method to give CrimeActivity control of creating the intents that other activities shall send to it

Reminder: Called by SingleFragmentActivity.onCreate to determine the fragment to be associated with the activity

Create and return the fragment to be associated with the activity

Retrieve ID of item to display from intent

Creating the Detail Fragment for an Item (For Display in One- or Two Pane Layout)

This is how data is passed to and received by a fragment: through fragment arguments

```
public class CrimeFragment extends Fragment {  
  
    private static final String ARG_CRIME_ID = "crime_id";  
    private Crime mCrime;  
  
    public static CrimeFragment newInstance(UUID crimeId) {  
        Bundle args = new Bundle();  
        args.putSerializable(ARG_CRIME_ID, crimeId);  
        CrimeFragment fragment = new CrimeFragment();  
        fragment.setArguments(args);  
        return fragment;  
    }  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        UUID crimeId = (UUID) getArguments().getSerializable(ARG_CRIME_ID);  
        mCrime = CrimeLab.get(getActivity()).getCrime(crimeId);  
    }  
  
    // ...
```

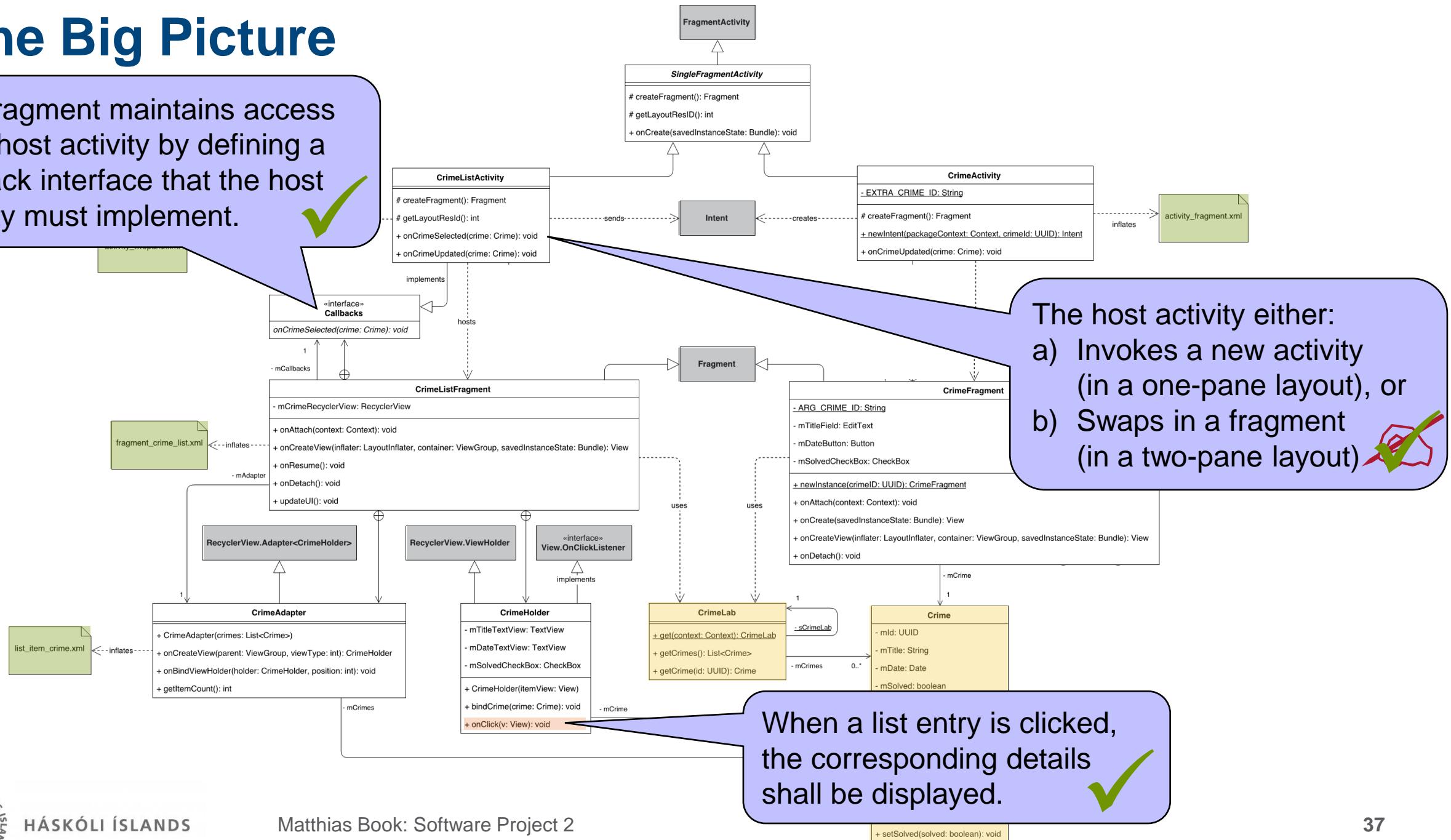
Static factory method

Place ID of item to show in an arguments bundle for the new fragment

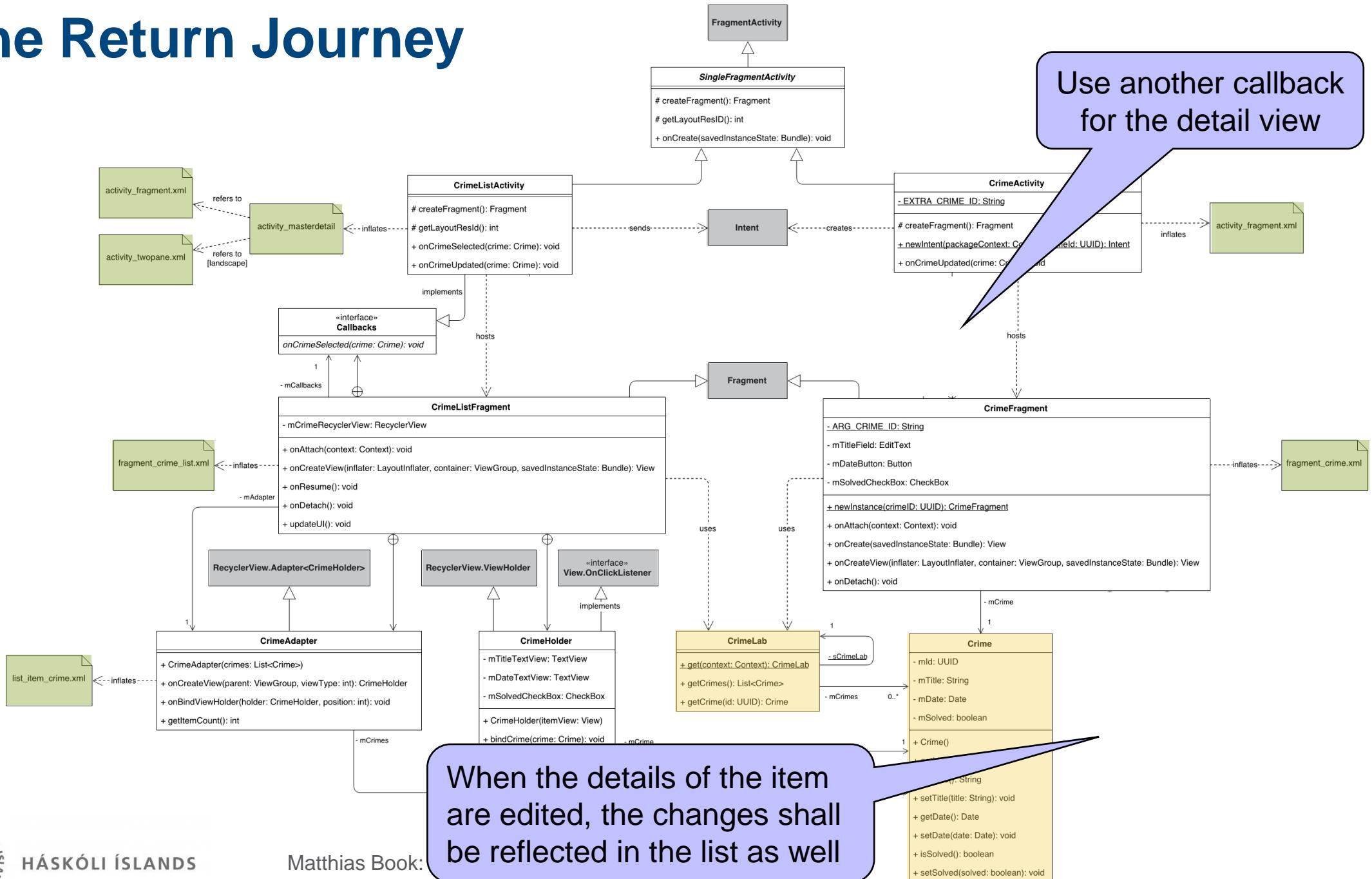
Retrieve ID of item to show from fragment arguments and store in member variable

The Big Picture

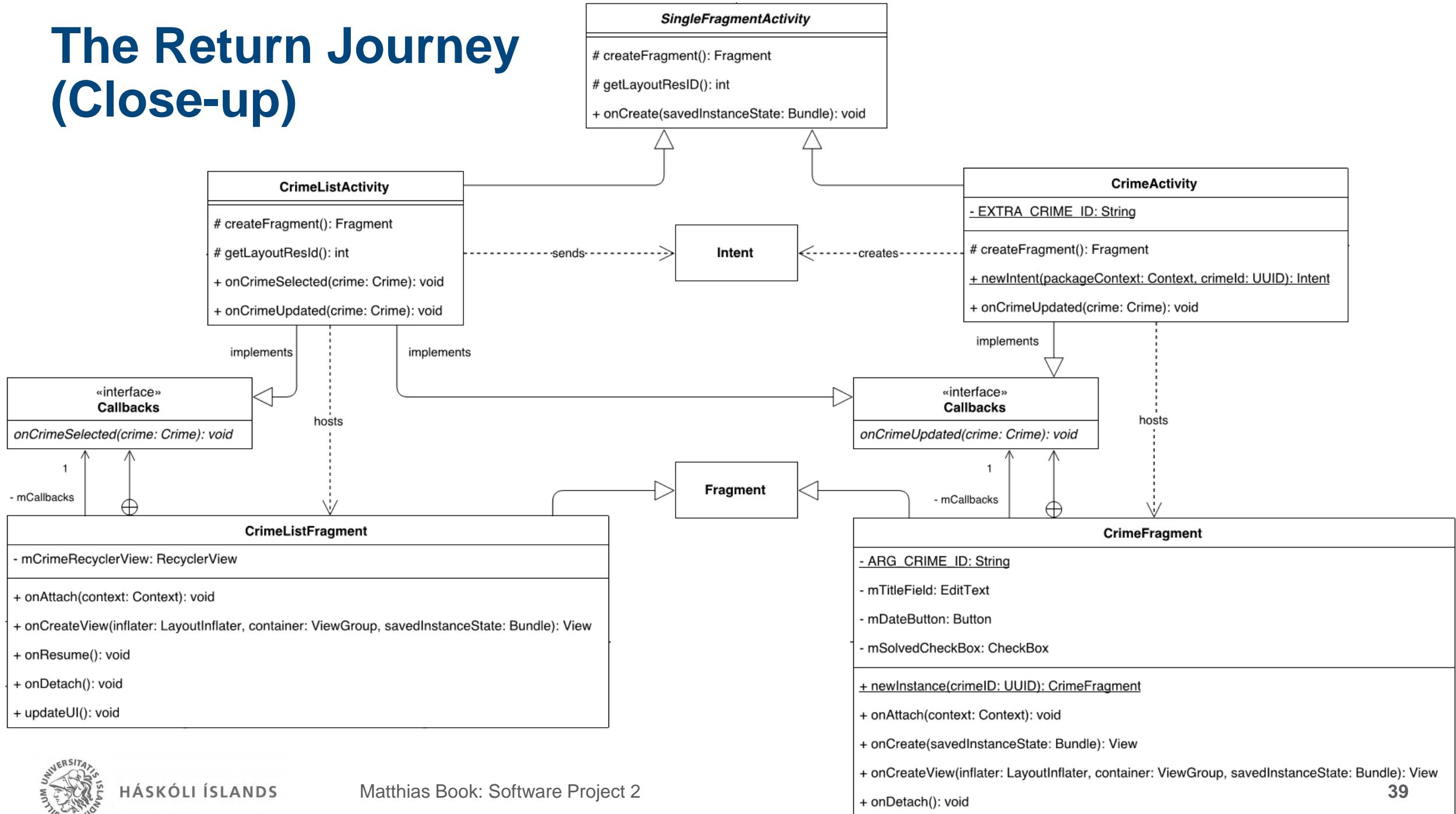
The fragment maintains access to its host activity by defining a callback interface that the host activity must implement.



The Return Journey



The Return Journey (Close-up)



Declaring the Detail's Callback Interface: CrimeFragment.Callbacks

```
public class CrimeFragment extends Fragment {  
  
    Reference to host activity  
  
    Declaration of interface that the  
    host activity must implement to  
    react to updates of detail data  
  
    Automatically stores a reference  
    to the host activity when the  
    fragment is associated with it.  
    ▪ In one-pane layout, this is  
      CrimeActivity  
    ▪ In two-pane layout, this is  
      CrimeListActivity  
  
    private Callbacks mCallbacks;  
  
    public interface Callbacks {  
        void onCrimeUpdated(Crime crime);  
    }  
  
    @Override  
    public void onAttach(Context context) {  
        super.onAttach(context);  
        mCallbacks = (Callbacks) context;  
    }  
  
    @Override  
    public void onDetach() {  
        super.onDetach();  
        mCallbacks = null;  
    }  
  
    // ...  
}
```



Invoking Callback in CrimeFragment's Event Listeners

```
public class CrimeFragment extends Fragment {  
    private Crime mCrime;  
    private CheckBox mSolvedCheckBox;  
    private Callbacks mCallbacks;  
    // ...  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
        Bundle savedInstanceState) {  
        View v = inflater.inflate(R.layout.fragment_crime, container, false);  
        // ...  
        mSolvedCheckBox = (CheckBox) v.findViewById(R.id.crime_solved);  
        mSolvedCheckBox.setChecked(mCrime.isSolved());  
        mSolvedCheckBox.setOnCheckedChangeListener(  
            new CompoundButton.OnCheckedChangeListener() {  
                @Override  
                public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {  
                    mCrime.setSolved(isChecked);  
                    mCallbacks.onCrimeUpdated(mCrime);  
                }  
            }  
        );  
        // ...  
    }  
    // ...  
}
```

This is how a fragment sends messages to another activity...

Example:
The checkbox event handler

Notify list activity (via callback reference) of update

Update the model

Implementing the Callback Interface: Crime[List]Activity.onCrimeUpdated

...which can then trigger logic in the fragment it hosts

```
public class CrimeListActivity extends SingleFragmentActivity
    implements CrimeListFragment.Callbacks, CrimeFragment.Callbacks {

    @Override
    public void onCrimeUpdated(Crime crime) {
        CrimeListFragment listFragment = (CrimeListFragment)
            getSupportFragmentManager().findFragmentById(R.id.fragment_container);
        listFragment.updateUI();
    }
    // ...
}
```

...and update its UI

In two-pane layout,
retrieve the list fragment
this activity is hosting...

```
public class CrimeActivity extends SingleFragmentActivity
    implements CrimeFragment.Callbacks {
    // ...
    @Override
    public void onCrimeUpdated(Crime crime) { }
}
```

In one-pane layout, the hosting fragment is
CrimeActivity, which does not have a list to update –
so this method implementation remains empty.



Ensuring the List is Updated in One-Pane Layout

```
public class CrimeListFragment extends Fragment {  
  
    private CrimeAdapter mAdapter;  
  
    @Override  
    public void onResume() {  
        super.onResume();  
        updateUI();  
    }  
  
    public void updateUI() {  
        CrimeLab crimeLab = CrimeLab.get(getActivity());  
        List<Crime> crimes = crimeLab.getCrimes();  
  
        if (mAdapter == null) {  
            mAdapter = new CrimeAdapter(crimes);  
            mCrimeRecyclerView.setAdapter(mAdapter);  
        } else {  
            mAdapter.notifyDataSetChanged();  
        }  
    }  
}  
  
// ...
```

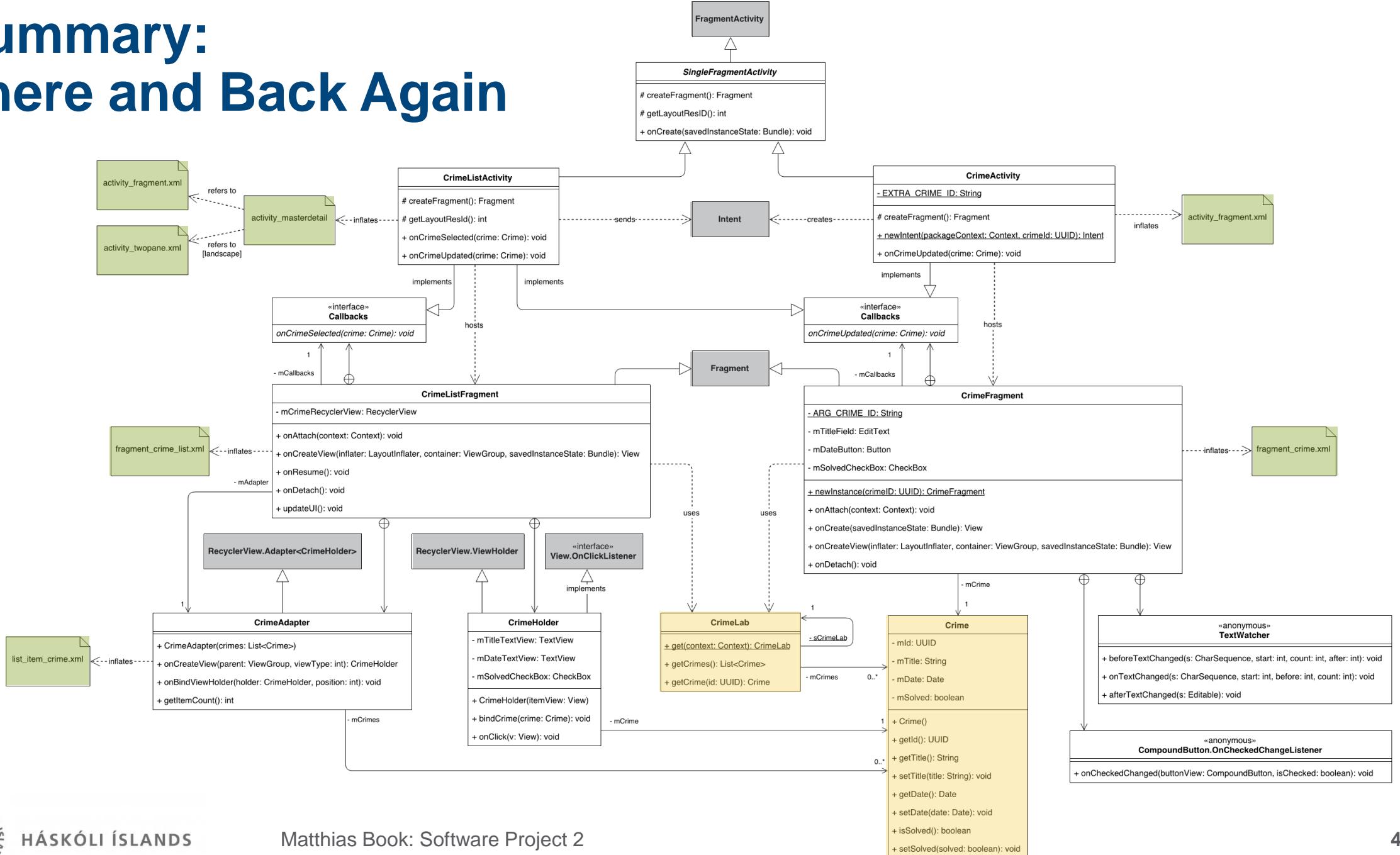
In one-pane layout, update the list UI when the list activity (and thus the list fragment) comes to the front again (i.e. resumes) after the user backed out of the detail view

To update the UI in one- and two-pane layout, retrieve current list of crimes from the model...

Caution – inefficient solution (chosen here for clarity): It would not be necessary to retrieve all list items again since just one item has been changed!

...and update the scrollable list with it (see Android textbook, Ch. 8 for details)

Summary: There and Back Again



In-Class Quiz 8: Activity & Fragment Communication

Note the numbers of the matching sentence parts:



- a) An activity can start another activity...
 - 1. ...by calling a method of a callback interface defined by the fragment and implemented by the activity.
 - 2. ...by calling a method of the fragment via the reference the activity holds to it.
 - 3. ...by creating an intent and passing it to the ActivityManager, who can then start the desired activity.
 - 4. ...by sending a message to the fragment's host activity, who can then start the desired activity.
 - 5. ...by sending a message to the fragment's host activity, who can then send a message to the other fragment.
 - 6. ...by telling the FragmentManager to start the desired fragment in a container in the activity's layout.
- b) An activity can start a fragment...
 - 1. ...by calling a method of a callback interface defined by the fragment and implemented by the activity.
 - 2. ...by calling a method of the fragment via the reference the activity holds to it.
 - 3. ...by creating an intent and passing it to the ActivityManager, who can then start the desired activity.
 - 4. ...by sending a message to the fragment's host activity, who can then start the desired activity.
 - 5. ...by sending a message to the fragment's host activity, who can then send a message to the other fragment.
 - 6. ...by telling the FragmentManager to start the desired fragment in a container in the activity's layout.
- c) A fragment can start an activity...
 - 1. ...by calling a method of a callback interface defined by the fragment and implemented by the activity.
 - 2. ...by calling a method of the fragment via the reference the activity holds to it.
 - 3. ...by creating an intent and passing it to the ActivityManager, who can then start the desired activity.
 - 4. ...by sending a message to the fragment's host activity, who can then start the desired activity.
 - 5. ...by sending a message to the fragment's host activity, who can then send a message to the other fragment.
 - 6. ...by telling the FragmentManager to start the desired fragment in a container in the activity's layout.
- d) An activity can send a message to a fragment it hosts...
 - 1. ...by calling a method of a callback interface defined by the fragment and implemented by the activity.
 - 2. ...by calling a method of the fragment via the reference the activity holds to it.
 - 3. ...by creating an intent and passing it to the ActivityManager, who can then start the desired activity.
 - 4. ...by sending a message to the fragment's host activity, who can then start the desired activity.
 - 5. ...by sending a message to the fragment's host activity, who can then send a message to the other fragment.
 - 6. ...by telling the FragmentManager to start the desired fragment in a container in the activity's layout.
- e) A fragment can send a message to the activity hosting it...
 - 1. ...by calling a method of a callback interface defined by the fragment and implemented by the activity.
 - 2. ...by calling a method of the fragment via the reference the activity holds to it.
 - 3. ...by creating an intent and passing it to the ActivityManager, who can then start the desired activity.
 - 4. ...by sending a message to the fragment's host activity, who can then start the desired activity.
 - 5. ...by sending a message to the fragment's host activity, who can then send a message to the other fragment.
 - 6. ...by telling the FragmentManager to start the desired fragment in a container in the activity's layout.
- f) A fragment can send a message to another fragment...
 - 1. ...by calling a method of a callback interface defined by the fragment and implemented by the activity.
 - 2. ...by calling a method of the fragment via the reference the activity holds to it.
 - 3. ...by creating an intent and passing it to the ActivityManager, who can then start the desired activity.
 - 4. ...by sending a message to the fragment's host activity, who can then start the desired activity.
 - 5. ...by sending a message to the fragment's host activity, who can then send a message to the other fragment.
 - 6. ...by telling the FragmentManager to start the desired fragment in a container in the activity's layout.





Hugbúnaðarverkefni 2 / Software Project 2

11. Software Architecture

HBV601G – Spring 2020

Matthias Book



HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRVÍSINDASVIÐ
ÍÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD

Participant View (Zoom App Window)



Matthias Book

Guðrún

Mic/cam (un)mute
Hold SPACE bar to unmute microphone temporarily for questions.

Unavailable in lecture; may be used in consultations:

Share screen

Record session

Enter breakout room

Quick feedback

Exit room

Speaker View

Participants (3)

| | |
|----------------------|----------|
| Jón (Me) | [Unmute] |
| Matthias Book (Host) | [Unmute] |
| Guðrún | [Unmute] |

Raise Hand yes no go slower go faster more

Unmute Me

Zoom Group Chat

From Matthias Book to Everyone:
Today's quiz is at
<https://forms.gle/0.....>

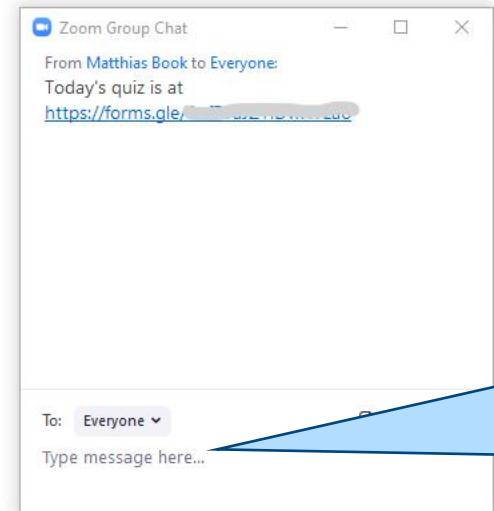
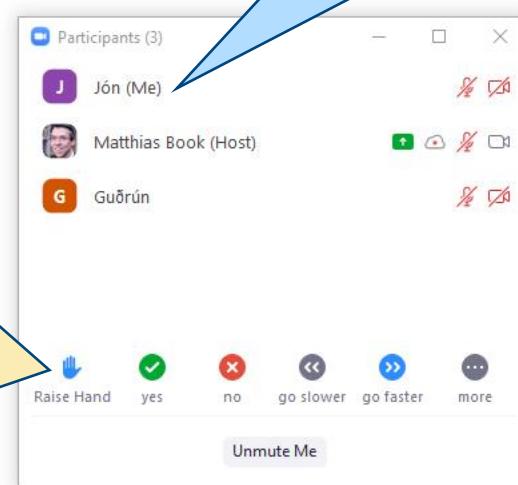
Screensharing View (Fullscreen)

Recording indicator: Lectures will likely be recorded, consultations won't.

Nonverbal feedback:

- Use “Raise Hand” in lectures to gain teacher’s attention.
- Teacher may not notice other feedback types in large lectures.

Hover to change your screen name



Use only in consultations

Switch between:

- Speaker name
- Speaker video
- Participant gallery

Public/private chat:
Teacher may miss messages, and can't conveniently read them during lectures.

Mic/cam (un)mute.
Hold SPACE bar to unmute microphone temporarily for questions.

Unavailable in lecture; may be used in consultations:

Share screen

Record session

Enter breakout room

Quick feedback

Exit room

Screensharing View (Zoom App Window)

Recording indicator:
Lectures will likely be recorded, consultations won't.

Participant Gallery

Nonverbal feedback:

- Use “Raise Hand” in lectures to gain teacher’s attention.
- Teacher may not notice other feedback types in large lectures.

Public/private chat:
Teacher may miss messages, and can’t conveniently read while screensharing.

Mic/cam (un)mute.
Hold SPACE bar to unmute microphone temporarily for questions.

Unavailable in lecture; may be used in consultations:

- Share screen
- Record session
- Enter breakout room
- Quick feedback
- Exit room

Participants (3)

- Jón (Me)
- Matthias Book (Host)
- Guðrún

Controls:

- Raise Hand
- yes
- no
- go slower
- go faster
- more

Unmute Me

Zoom Group Chat

From Matthias Book to Everyone:
Today's quiz is at
<https://forms.gle/>

Bottom Bar:

- Unmute
- Start Video
- Invite
- Participants (3)
- Share Screen
- Chat (highlighted with red circle)
- Record
- Breakout Rooms
- Reactions
- Leave Meeting

To: Everyone
Type message here...
File
4

Privacy in Zoom Teleconferences and Recordings

- While your microphone is on (unmuted), your **audio** will be broadcast to other participants and included in the lecture recording.
- While your camera is on, your **video** will be visible to other participants in the Participant Gallery. Your video may be included in the lecture recording while you are speaking, or while the teacher is not screensharing.
- While your camera is off, your **screen name** is visible to other participants in the Participant Gallery, and may be visible in the recording while you are speaking, or while the teacher is not screensharing.
- **Chat messages** are visible to the addressed participants (incl. possibly everyone) and may be saved by them. They are not visible in the recording.
- **Privacy notice:** By activating your microphone or camera or chatting, you consent to your audio/video/text being stored as part of the lecture recording on Zoom and Panopto servers, and being broadcast to other participants live and when playing back the lecture recording. The teachers' lecture recordings are not public, but are only made accessible to your classmates. Participants will generally not be given permission to record lectures themselves, but may record consultations if the teacher and all participants agree.



Switch to Remote Teaching due to COVID-19: Lectures

- Classroom lectures discontinued as of this week
- Lectures proceed via **teleconference** for rest of semester
 - Suggestion: Start at original time of 08:20 already, so we can take a break in the middle?
 - Zoom conference room for all HBV601G lectures: <https://eu01web.zoom.us/j/775961381>
 - Need to download small client software (first time only)
 - Upon entering, your microphone will be muted and your camera off by default
 - To ask a question, use “Raise Hand” function
 - Hold SPACE bar to unmute temporarily as you speak
 - In-class quiz responses submitted via Google Forms (only answers on lecture day count)
- Lecture **recordings** and **slides** will be available as usual
 - Accessible via Uglar or <https://rec.hi.is>
 - Please log in with your Uglar account (Moodle accounts don't have access)



Switch to Remote Teaching due to COVID-19: Consultations

- **Classroom** consultations discontinued as of last week
- Consultations proceed via **teleconference** for rest of semester
 - Please schedule consultation and presentation times with your tutors
 - Use your regular consultation timeslots
 - For Assignment 3 presentations, schedule times together with your Code Review partner team
 - Tutors will send you telco invitation links
 - One team member shares their screen
 - Camera usage encouraged
- Remote final presentation format to be announced later



Switch to Remote Teaching due to COVID-19: Projects

- Physical team meetings discouraged due to social distancing recommendations
- Strongly encouraged to schedule fixed teleconferences with teammates
 - Suggested teleconferencing tools supporting screen sharing:
 - <https://meet.jit.si> (free & unlimited, runs in browser – best in Chrome, others unstable)
 - <https://whereby.com> (free for meetings of up to 4 participants, runs in browser)
 - <https://zoom.us> (free 1:1 meetings, free 40-minute meetings for 3+ people)
- Get help with questions on Piazza
 - <https://piazza.com/hi.is/spring2020/hbv601g/>
- *Remote teaching feedback and suggestions appreciated via book@hi.is!*



In-Class Quiz 9 Prep

- During class, I'll show you questions that you can answer with a number
- **New:** Submit your quiz via Google Forms
- All responses submitted on the day of the lecture count
- All questions in a quiz have same weight
- All quizzes (8-10 throughout semester) have the same weight
 - Your worst 2 quizzes will be disregarded
- Overall quiz grade counts as optional question worth 7.5% on final exam



In-Class Quiz 8: Activity & Fragment Communication Solution



- a) An activity can start another activity (3) by creating an intent and passing it to the ActivityManager, in order to let it start the desired activity.
- b) An activity can start a fragment (6) by telling the FragmentManager to place the desired fragment in a container in the activity's layout.
- c) A fragment can start an activity (4) by sending a message to the fragment's host activity, who can then invoke the desired activity.
- d) An activity can send a message to a fragment it hosts (2) by calling a method of the fragment via the reference the activity holds to it.
- e) A fragment can send a message to the activity hosting it (1) by calling a method on a callback interface defined by the fragment and implemented by the activity.
- f) A fragment can send a message to another fragment (5) by sending a message to the fragment's host activity, who can then send a message to the other fragment.

Preview: Assignment 4: Schedule and Deliverables

- On **Thu 11 Apr**, demonstrate and **explain** your product to your classmates in a 10- to 15-minute-presentation:
 1. **Product:** What does your product do? Demonstrate the key features of your system.
 2. **Architecture:** How does your product work? Explain architecture & key design decisions.
 3. **Process:** How did you build the product? Relate and interpret challenges you faced.
- On **Sun 14 Apr**, submit your **final product** in Ugla, incl.:
 - Complete source code and installation instructions
 - Slides of your final presentation
- Your product does not need to satisfy all the criteria you listed in your initial requirements document, but the key features should work.

Presentation delivery format
(video, telco, etc.) and
schedule to be announced



Assignment 4: Grading

- Assignment (Team) grade refers to the visible parts of the presentation
 - i.e. information on slides, impression the product makes, etc. – Criteria:
 - Final product implements key features, and is running smoothly (75%)
 - Critical retrospective given of chosen process, architecture and technology (25%)
- Presentation grade refers to the audible parts of the presentation
 - Criteria: how well the presenter explains things, answers questions etc.
 - Presentation to tutor must be given by team member who has not presented any assignments yet (other presentations, if any, may be given by other team members)
 - If all team members have gotten a presentation grade already, the presentation quality will be reflected in the team grade.



Software Architecture

see also:

Larman: Applying UML and Patterns, Ch. 13 & 33



HÁSKÓLI ÍSLANDS

Matthias Book: Software Project 2



Definition: Software Architecture

“A software architecture is

- the set of significant **decisions** about the **organization** of a software system,
- the **selection of the structural elements** of which the system is composed,
 - and their **interfaces**,

together with

- their behavior, as specified in the **collaborations** among those elements,
- the **composition** of these structural and behavioral elements
 - into progressively larger subsystems,

and

- the architectural **style** that guides this organization
 - of the elements and their interfaces, their collaboration, and their composition.”



Architectural Analysis

- How do we come up with a software architecture?
- **The essence of architectural analysis** is to
 - **identify factors** that should influence the architecture,
 - **understand** their **variability** and **priority**, and
 - **resolve** them by making architectural **decisions**.
- Challenge: It's difficult to...
 - Know what questions to ask
 - Weigh the trade-offs
 - Know the many ways to resolve an architecturally significant factor
 - Decide on the best way under the given circumstances



Examples of Issues to be Resolved at Architectural Level

- How do **reliability** and **fault-tolerance** requirements affect the **design**?
 - For what remote services will fail-over to local services be allowed? Why?
 - Do the local and remote services work exactly the same? What are the differences?
- How do the **licensing** costs of purchased subcomponents affect **profitability**?
 - Should we integrate a high-quality third-party persistence framework that will charge a fee on all transactions, go with a low-cost open-source alternative, or develop our own?
- How do the **adaptability** and **configurability** requirements affect the **design**?
 - What variations of business rules need to be reflected in the implementation?
 - What degree of evolution should be accommodated? How should variations be specified?
- How do **availability** and **dependability** requirements influence the **effort**?
 - Very strict availability requirements may induce huge costs for redundant hardware, hot-swap capability, failover mechanisms, backups, etc. Is this effort commensurate with the risk?

Common Steps in Architectural Analysis

- Goal: Understand **influence**, **priorities** and **variability** of architectural drivers

1. Identify and analyze **architectural drivers***, i.e. requirements that have an architectural impact

- Especially non-functional (quality) requirements
- But also functional requirements, esp. regarding expected variability or change
- Can be found e.g. in
 - Vision and Scope document
 - Business Rules document
 - Supplementary Specification document
 - Use Case document (sections on special requirements, technology variations, open issues in fully-dressed format)

2. Make **architectural decisions**

- Analyze alternatives
- Create solutions that address the impact:
 - Build a custom solution
 - Buy a third-party solution
 - Remove the requirement
 - Hire an expert
 - ...

3. Document decisions

- So people will not inadvertently undermine design decisions later
- or spend time pursuing rejected options



Making Architectural Decisions

- **Collecting and describing** architectural drivers is comparatively easy.
- **Addressing and resolving** them in light of **interdependencies** and **trade-offs** is much more difficult.
 - Highly dependent on application domain and technology
 - Business goals and stakeholder interests become central to technical decisions
 - Requires consideration of more large-scale/global goals and trade-offs than local-scale UI or OO design decisions
 - Requires knowledge and critical consideration of many areas:
 - Architectural styles and patterns, technologies, products, pitfalls, domain knowledge, trends...
- Hierarchy of **goals** to guide **priority** of architectural decisions:
 1. Inflexible constraints, e.g. safety and legal compliance – unavoidable
 2. Business goals, e.g. particular features, deadlines etc. – some flexibility
 3. All other goals (are often derived from business goals) – some leeway for interpretation



Characteristics of Software Architecture

- Architectural concerns are especially **related to non-functional requirements**, but their resolution **permeates the implementation of the functional requirements**.
- Architectural concerns **require awareness of the business context, stakeholder goals**, as well as **requirements' variability and evolution**.
- Architectural concerns involve **system-level, large-scale issues** whose resolution usually involves **large-scale, fundamental decisions** that are **extremely costly to change** later on.
- Architectural decisions depend on the **conception** and **critical evaluation** of **alternative solutions**.
- Architectural decisions usually involve **interdependencies** and **trade-offs**.

Architectural Structures

see also:

- Bass et al.: Software Architecture in Practice, Ch. 1-3



Definition: Architectural Structures

= “Perspectives” on a system

Can be described in design documents, UML diagrams, etc.

- Software architecture is the set of structures needed to **reason** about a system.
- A structure is a set of software **elements** and the **relations** between them.
 - **Module structures** show how a system is to be statically structured as a set of code or data units that have to be constructed or procured.
 - **Component-and-connector structures** show how a system is to be dynamically structured as a set of modules having runtime behavior (components) and interactions (connectors).
 - **Allocation structures** show how software structures are to be mapped to the system’s organizational, developmental, installation and execution environments.
- Each structure has the potential to **influence quality attributes** of the system.
 - e.g. availability, interoperability, modifiability, performance, security, testability, usability
- A system’s architecture is **established through a series of design decisions**.
 - e.g. responsibility allocation, resource management, binding times, technology choices
- These **occur in** and **affect** technical, project, business, professional **contexts**.

Definition: Architectural Structures

- Software architecture
- A structure is a set of design decisions
 - **Module structures** show how a system is to be structured as a set of code or data units that interact.
 - **Component-and-connector structures** show how a system is to be structured as a set of modules having runtime behavior (components) and interactions (connectors).
 - **Allocation structures** show how software structures are to be mapped to the system's organizational, developmental, installation and execution environments.
- Each structure has the potential to **influence quality attributes** of the system.
 - e.g. availability, interoperability, modifiability, performance, security, testability, usability
- A system's architecture is **established through a series of design decisions**.
 - e.g. responsibility allocation, resource management, binding times, technology choices
- These **occur in and affect** technical, project, business, professional **contexts**.

“Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled.”

Eoin Woods



Module Structures

- Module structures show how a system is to be **statically structured** as a **set of code or data units** that have to be constructed or procured.
- Focus on modules and relationships established at **design time**
 - e.g. layers such as database, business logic, user interface; subdivided further into components, which are subdivided into classes, which are subdivided into methods
 - **Which modules are there? How are they composed? How do they rely on each other?**
- Examples of module structures:
 - **Decomposition structure:** Shows how modules are recursively composed of submodules
 - Relevant for localizing responsibilities expressed in functional requirements, organizing the project, supporting modifiability through encapsulation, etc.
 - **Uses structure:** Shows how modules are depending on other modules
 - Relevant for setting priorities, tracing faults, supporting extensibility and reusability, etc.

Component-and-Connector Structures

- Component-and-connector structures show how the modules **behave and interact** with each other at run time to carry out the system's **functions**.
- Focus on module instances and their relationships manifested at **run time**
 - i.e. components such as services, peers, clients, servers, filters etc., and connectors such as call-returns, process synchronization operations, pipes, etc.
 - **How are the module instances hooked together at runtime? How do they interact?**
- Examples of component-and-connector structures:
 - **Service structure:** Shows how (possibly independently engineered) services can interoperate with each other using a service coordination mechanism
 - Relevant for supporting interoperability, reliability etc.
 - **Concurrency structure:** Shows where opportunities for parallelism exist and where resource contention may occur
 - Relevant for supporting performance, testability, availability etc.

Allocation Structures

- Allocation structures show how **software structures** are to be **mapped to the system's** organizational, developmental, installation & execution **environments**.
- Focus on non-software structures in the system's **environment**
 - e.g. devices, CPUs, file systems, networks, development teams, etc.
 - **Who builds the modules? Where are they run/stored? What infrastructure do they use?**
- Examples of allocation structures:
 - **Work assignment structure:** Shows how responsibility for implementing the modules is assigned to teams, and which expertise is required on each team
 - Relevant for staffing, project management, collaboration infrastructure etc.
 - **Deployment structure:** Shows how modules are allocated to hardware components for processing and communication, and how they may migrate between hardware components
 - Relevant for supporting performance, data integrity, security and availability, especially in distributed and parallel systems

Design Decisions Shaping Architectural Structures

- A **module structure** defines e.g.:
 - What is the primary functional responsibility assigned to each module?
 - What other modules is a module allowed to use?
 - What other modules does it actually use and depend on?
 - What modules are related to other modules by inheritance relationships?
- An **allocation structure** defines e.g.:
 - What processor is each module instance executed on?
 - In what directories or files is each module stored during development, testing and building?
 - What is the assignment of each module to development teams?
- A **component-and-connector structure** defines e.g.:
 - What are the major executing modules and how do they interact at runtime?
 - What are the major shared data stores?
 - Which parts of the system are replicated?
 - How does data progress through the system?
 - What parts of the system can run in parallel?
 - Can the system's structure change as it executes, and if so, how?
 - What run-time properties (performance, security, availability etc.) does the system exhibit?



Architectural Design Decisions



see also:

- Bass et al.: Software Architecture in Practice, Part II



HÁSKÓLI ÍSLANDS

Matthias Book: Software Project 2

Recap: Architecture's Impact on Quality Attributes

- A system's architecture **determines** the system's **quality attributes**, e.g.
 - Availability
 - Interoperability
 - Modifiability
 - Performance
 - Security
 - Testability
 - Usability
- A system's architecture is **established** in a series of **design decisions**, e.g. on
 - Allocation of responsibilities
 - Coordination model
 - Data model
 - Management of resources
 - Mapping of architectural elements
 - Variations and binding times
 - Technology choices



Architectural Design Decisions

Shaping the Module Structure

- Decisions about the **allocation of responsibilities**
(i.e. where functional requirements will manifest themselves)
 - Identifying the important responsibilities, including basic system functions, architectural infrastructure, satisfaction of quality attributes
 - Determining how these responsibilities are allocated to modules at design time and run time
- Decisions about the **data model**
(i.e. the representation of entities whose processing is the main purpose of the system)
 - Choosing the major data abstractions, their operations and properties
 - i.e. how to create, initialize, access, persist, manipulate, translate, and destroy data entities
 - Compiling metadata needed for consistent interpretation of the data
 - Organizing the data
 - e.g. object-oriented vs. relational representation, conversion between both representations

Architectural Design Decisions

Shaping the Component-and-Connector Structure

- Decisions about the **coordination model**
(i.e. how modules interact through designed mechanisms)
 - Identifying the modules that must cooperate, or are prohibited from cooperating
 - Determining the properties of the cooperation, e.g. timeliness, currency, consistency etc.
 - Choosing the communication mechanisms between modules and systems
 - e.g. stateful vs. stateless, synchronous vs. asynchronous, guaranteed vs. best-effort
- Decisions about **variations** and **binding times**
(i.e. allowable range, time and mechanism of variations of a system)
 - Identifying variation points in which functionality/properties of the system can be changed
 - Deciding on the range of supported variations per variation point
 - Deciding on the binding time of variations, and the actor choosing a variation
 - e.g. at design time by the developer, at deploy time by an install wizard, at run time by the user...
 - Choosing mechanisms to specify and execute the variation
 - e.g. in source code, compiler directives, make files, configuration files, graphical user interface...

Architectural Design Decisions

Shaping the Allocation Structure

- Decisions about the **mapping between architectural elements**

(i.e. between elements of development and execution; and software and hardware elements)

- Mapping design-time modules and run-time instances
- Assigning run-time elements to devices or processors
- Assigning entities in the data model to data stores

- Decisions about **management of resources**

(i.e. arbitrating the use of shared resources, e.g. CPU, storage, peripherals etc.)

- Identifying resources to be managed, and determining limits for each
- Determining which modules should manage which resource
- Determining how resources are shared, and what arbitration mechanisms are required
- Determining and avoiding the impacts of saturation of different resources



Architectural Design Decisions

Shaping All Structures

- Decisions about **technology choices**

(i.e. selection of suitable technologies to support preceding architectural decisions)

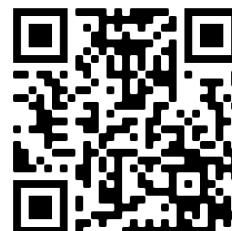
- Identifying candidate technologies that can realize decisions made in preceding categories
- Determining whether adequate internal and external expertise is available for a technology
- Determining side effects of a technology, e.g. in coordination or resource management
- Determining compatibility of a new technology with existing technology stack
- Determining whether available tools adequately support technology choices
- Choosing a suitable technology to support a particular requirement or architectural decision

In-Class Quiz #9: Architectural Design Decisions



Decisions about...

- a) the allocation of responsibilities
- b) the coordination model
- c) the data model
- d) the management of resources
- e) the mapping between architectural elements
- f) technology choices
- g) variations and binding times



...determine:

Answer at <https://forms.gle/C9LqbZ9oGYzYTP9M9>

1. the arbitration of the use of shared resources, e.g. CPU, storage, peripherals etc.
2. the representation of entities whose processing is the main purpose of the system
3. the association of elements of development and execution; and of software and hardware elements
4. where functional requirements will manifest themselves
5. how modules interact through designed mechanisms
6. the allowable range, time and mechanism of variations of a system
7. the selection of suitable technologies to support the above architectural decisions



Hugbúnaðarverkefni 2 / Software Project 2

12. Software Architecture

HBV601G – Spring 2020

Matthias Book



HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRVÍSINDASVIÐ
ÍÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD

Further Questions / Ongoing Discussion

- **New department-wide Piazza forum:**
<https://piazza.com/hi.is/spring2020/csdept>
- For discussion of teaching...
 - methods
 - tools
 - policies
 - logistics
 - exams
 - etc.
- ...during the time of the assembly ban
- Support forum with optional participation – **please sign up yourself!**



Changes in Course Assessment

- Goals, to the degree possible:
 - Maintain original course structure and content...
 - to provide you planning security, methodical familiarity, and maintain exam archive's usefulness
 - ...while giving you flexibility on the scope of the assessment
 - to reflect disruptions of studies due to assembly ban, additional family responsibilities etc.
- Types and weights of assessment remain as originally announced, but expectations on content and delivery are relaxed



Changes in Course Assessment

- **In-class quizzes**
 - To be submitted **anytime** on lecture day instead of **during lecture**
 - Worst **three** instead of **two** quizzes are disregarded
- **Assignments**
 - Submission deadlines on **Monday** instead of **Sunday** nights
- **Peer assessment**
 - Please focus on contributions **before** rather than **since** assembly ban
- **Final exam**
 - Digital take-home exam instead of **on-campus paper** exam
 - Open-book instead of **closed-book** exam
 - Extra 30 min of exam time for taking breaks
 - Worst **two** questions are disregarded (instead of **all mandatory**)
 - Teacher available for questions via Zoom **anytime** instead of **two visits**



In Exchange...

- **No collaboration**, please – you're on your honor!
- I will place stronger emphasis on answers being **in your own words**.
 - Answers that mirror phrases on my slides will receive reduced points.
 - Answers that mirror other students' answers will receive zero points.
- If I have the strong impression that parts of an exam are not your own work, I reserve the right to invite you to an oral exam to determine the exam grade.
- **Don't stress out** about these rules – if you're working on your own, you're fine.



Discussion

What are your

- Thoughts
- Suggestions
- Questions
- Concerns
- Feedback



▪ Ask via audio

- Hold SPACE bar to unmute microphone temporarily while you are speaking

▪ Ask via chat

- Send questions to “Everyone”



Assignment 4: Schedule and Deliverables

- On **Thu 16 Apr**, demonstrate and **explain** your product to your classmates in a 10- to 15-minute-presentation:
 1. **Product:** What does your product do? Demonstrate the key features of your system.
 2. **Architecture:** How does your product work? Explain architecture & key design decisions.
 3. **Process:** How did you build the product? Relate and interpret challenges you faced.
- On **Mon 20 Apr**, submit your **final product** in Uglar, incl.:
 - Complete source code and installation instructions
 - Slides of your final presentation
- Your product does not need to satisfy all the criteria you listed in your initial requirements document, but the key features should work.

Presentation delivery format
(video, telco, etc.) and
schedule to be announced
next week



Recap: Assignment 4: Grading

- Assignment (Team) grade refers to the visible parts of the presentation
 - i.e. information on slides, impression the product makes, etc. – Criteria:
 - Final product implements key features, and is running smoothly (75%)
 - Critical retrospective given of chosen process, architecture and technology (25%)
- Presentation grade refers to the audible parts of the presentation
 - Criteria: how well the presenter explains things, answers questions etc.
 - Presentation to tutor must be given by team member who has not presented any assignments yet (other presentations, if any, may be given by other team members)
 - If all team members have gotten a presentation grade already, the presentation quality will be reflected in the team grade.



Recap: Architectural Analysis

- How do we come up with a software architecture?
- **The essence of architectural analysis** is to
 - **identify factors** that should influence the architecture,
 - **understand** their **variability** and **priority**, and
 - **resolve** them by making architectural **decisions**.
- Challenge: It's difficult to...
 - Know what questions to ask
 - Weigh the trade-offs
 - Know the many ways to resolve an architecturally significant factor
 - Decide on the best way under the given circumstances



Recap: Architectural Structures

- An **architectural structure** is a set of software **elements** and the **relations** between them (*a “perspective” for reasoning about certain aspects of a system*)
 - **Module structures** show how a system is to be **statically structured** as a set of **code or data units** that have to be constructed or procured.
 - Allocation of responsibilities
 - Data model
 - **Component-and-connector structures** show how a system is to be **dynamically structured** as a set of modules having **runtime behavior** (components) and **interactions** (connectors).
 - Coordination model
 - Variations and binding times
 - **Allocation structures** show how software structures are to be **mapped** to the system’s (**non-software**) organizational, developmental, installation and execution environments
 - Mapping between architectural elements
 - Management of resources
 - **Crosscutting all structures:** Technology choices



In-Class Quiz 9 Solution: Architectural Design Decisions



- a) Decisions about the **allocation of responsibilities**
 - determine where functional requirements will manifest themselves (4)
- b) Decisions about the **coordination model**
 - determine how modules interact through designed mechanisms (5)
- c) Decisions about the **data model**
 - determine the representation of entities whose processing is the main purpose of the system (2)
- d) Decisions about **management of resources**
 - determine the arbitration of use of shared resources, e.g. CPU, storage, peripherals etc. (1)
- e) Decisions about the **mapping between architectural elements**
 - determine the association of elements of development and execution; and software and hardware elements (3)
- f) Decisions about **technology choices**
 - determine the selection of suitable technologies to support all these architectural decisions (7)
- g) Decisions about **variations and binding times**
 - determine allowable range, time and mechanism of variations of a system (6)

In-Class Quiz #10: Software Architecture

Answer on lecture day at

<https://forms.gle/wFKKNgswygWxwLYo9>

Fill in the blanks using the following:

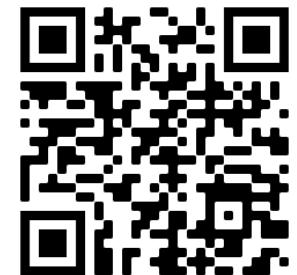
- (A)llocation Structure, (C)omponent-and-Connector Structure, (D)esign Decisions, (M)odule Structure, (Q)uality Attributes, (S)ystem Architecture, Conte(x)ts

- A (a) _____ can be considered from several perspectives:
 - (b) _____: Design-time view of divisions and relationships of constructed modules
 - (c) _____: Run-time view of interactions of module instances
 - (d) _____: Design- and run-time view of mapping architectural elements to each other

- A system's architecture is **established through a series of (e) _____.**
 - e.g. allocation of responsibilities, coordination model, data model, resource management, mapping of architectural elements, variations and binding times, technology choices

- Design decisions **influence (f) _____ of the system.**
 - e.g. availability, interoperability, modifiability, performance, security, testability, usability

- Decisions affect & depend on technical, project, business, professional (g) _____.



Architectural Tactics



see also:

- Bass et al.: Software Architecture in Practice, Part II



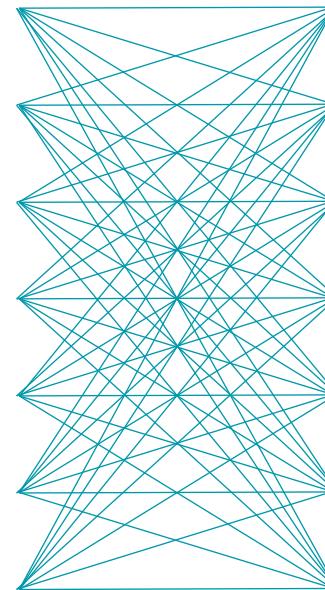
HÁSKÓLI ÍSLANDS

Matthias Book: Software Project 2

Architectural Tactics

Design Decisions

- Allocation of responsibilities
- Coordination model
- Data model
- Management of resources
- Mapping of arch. elements
- Variations and binding times
- Technology choices



Quality Attributes

- Availability
- Interoperability
- Modifiability
- Performance
- Security
- Testability
- Usability
- Variability
- Portability
- Scalability
- Mobility
- Safety
- Deployability
- Monitorability
- Shorability

- An **architectural tactic** is a design decision that affects a quality attribute.

Availability Tactics

- **Prevent faults**

- Removal from service
- Transactions
- Predictive model
- Exception prevention
- Increase competence set

- **Detect faults**

- Ping/echo
- Monitor
- Heartbeat
- Timestamp
- Sanity checking
- Condition monitoring
- Voting
- Exception detection
- Self-test

- **Recover from faults**

- **Preparation and repair**

- Active redundancy
- Passive redundancy
- Spare
- Exception handling
- Rollback
- Software upgrade
- Retry
- Ignore faulty behavior
- Degradation
- Reconfiguration

- **Reintroduction**

- Shadow
- State resynchronization
- Escalating restart
- Non-stop forwarding



Modifiability Tactics

- **Reduce module size**
 - Split module
- **Increase cohesion**
 - Increase semantic coherence
- **Reduce coupling**
 - Encapsulate
 - Use an intermediary
 - Restrict dependencies
 - Refactor
 - Abstract common services
- **Defer binding**



Security Tactics

- **Detect attacks**

- Detect intrusion
- Detect service denial
- Verify message integrity
- Detect message delay

- **React to attacks**

- Revoke access
- Lock computer
- Inform actors

- **Resist attacks**

- Identify actors
- Authenticate actors
- Authorize actors
- Limit access
- Limit exposure
- Encrypt data
- Separate entities
- Change default settings

- **Recover from attacks**

- Maintain audit trail
- Restore (*see Availability*)



Testability Tactics

- **Control and observe system state**
 - Specialized interfaces
 - Record / playback
 - Localize state storage
 - Abstract data sources
 - Sandbox
 - Executable assertions
- **Limit complexity**
 - Limit structural complexity
 - Limit nondeterminism



Usability and Interoperability Tactics

Usability Tactics

- **Support user initiative**
 - Cancel
 - Undo
 - Pause / resume
 - Aggregate
- **Support system initiative**
 - Maintain task model
 - Maintain user model
 - Maintain system model

Interoperability Tactics

- **Locate**
 - Discover service
- **Manage interfaces**
 - Orchestrate
 - Tailor interface



Performance Tactics

- **Control resource demand**

- Manage sampling rate
- Limit event response
- Prioritize events
- Reduce overhead
- Bound execution times
- Increase resource efficiency

- **Manage resources**

- Increase resources
- Introduce concurrency
- Maintain multiple copies of computations
- Maintain multiple copies of data
- Bound queue sizes
- Schedule resources



Example: Design Checklist for Performance

| Category | Checklist |
|--------------------------------|---|
| Allocation of Responsibilities | <p>Determine the system's responsibilities that will involve heavy loading, have time-critical response requirements, are heavily used, or impact portions of the system where heavy loads or time-critical events occur.</p> <p>For those responsibilities, identify the processing requirements of each responsibility, and determine whether they may cause bottlenecks.</p> <p>Also, identify additional responsibilities to recognize and process requests appropriately, including</p> <ul style="list-style-type: none">▪ Responsibilities that result from a thread of control crossing process or processor boundaries▪ Responsibilities to manage the threads of control—allocation and deallocation of threads, maintaining thread pools, and so forth▪ Responsibilities for scheduling shared resources or managing performance-related artifacts such as queues, buffers, and caches <p>For the responsibilities and resources you identified, ensure that the required performance response can be met (perhaps by building a performance model to help in the evaluation).</p> |



Example: Design Checklist for Performance

| | |
|--------------------|---|
| Coordination Model | <p>Determine the elements of the system that must coordinate with each other—directly or indirectly—and choose communication and coordination mechanisms that do the following:</p> <ul style="list-style-type: none">▪ Support any introduced concurrency (for example, is it thread safe?), event prioritization, or scheduling strategy▪ Ensure that the required performance response can be delivered▪ Can capture periodic, stochastic, or sporadic event arrivals, as needed▪ Have the appropriate properties of the communication mechanisms; for example, stateful, stateless, synchronous, asynchronous, guaranteed delivery, throughput, or latency |
| Data Model | <p>Determine those portions of the data model that will be heavily loaded, have time-critical response requirements, are heavily used, or impact portions of the system where heavy loads or time-critical events occur.</p> <p>For those data abstractions, determine the following:</p> <ul style="list-style-type: none">▪ Whether maintaining multiple copies of key data would benefit performance▪ Whether partitioning data would benefit performance▪ Whether reducing the processing requirements for the creation, initialization, persistence, manipulation, translation, or destruction of the enumerated data abstractions is possible▪ Whether adding resources to reduce bottlenecks for the creation, initialization, persistence, manipulation, translation, or destruction of the enumerated data abstractions is feasible |



Example: Design Checklist for Performance

| Category | Checklist |
|--------------------------------------|---|
| Mapping among Architectural Elements | <p>Where heavy network loading will occur, determine whether co-locating some components will reduce loading and improve overall efficiency.</p> |
| Resource Management | <p>Ensure that components with heavy computation requirements are assigned to processors with the most processing capacity.</p> <p>Determine where introducing concurrency (that is, allocating a piece of functionality to two or more copies of a component running simultaneously) is feasible and has a significant positive effect on performance.</p> <p>Determine whether the choice of threads of control and their associated responsibilities introduces bottlenecks.</p> |
| | <p>Determine which resources in your system are critical for performance. For these resources, ensure that they will be monitored and managed under normal and overloaded system operation. For example:</p> |
| | <ul style="list-style-type: none"><li data-bbox="1205 951 2449 1052">▪ System elements that need to be aware of, and manage, time and other performance-critical resources<li data-bbox="1205 1052 2449 1095">▪ Process/thread models<li data-bbox="1205 1095 2449 1139">▪ Prioritization of resources and access to resources<li data-bbox="1205 1139 2449 1182">▪ Scheduling and locking strategies<li data-bbox="1205 1182 2449 1280">▪ Deploying additional resources on demand to meet increased loads |



Example: Design Checklist for Performance

Binding Time

For each element that will be bound after compile time, determine the following:

- Time necessary to complete the binding
- Additional overhead introduced by using the late binding mechanism

Ensure that these values do not pose unacceptable performance penalties on the system.

Choice of Technology

Will your choice of technology let you set and meet hard, real-time deadlines? Do you know its characteristics under load and its limits?

Does your choice of technology give you the ability to set the following:

- Scheduling policy
- Priorities
- Policies for reducing demand
- Allocation of portions of the technology to processors
- Other performance-related parameters

Does your choice of technology introduce excessive overhead for heavily used operations?



Example: Design Checklist for Modifiability

| Category | Checklist |
|--------------------------------|--|
| Allocation of Responsibilities | <p>Determine which changes or categories of changes are likely to occur through consideration of changes in technical, legal, social, business, and customer forces. For each potential change or category of changes:</p> <ul style="list-style-type: none"><li data-bbox="1128 318 2332 404">▪ Determine the responsibilities that would need to be added, modified, or deleted to make the change.<li data-bbox="1128 404 2332 491">▪ Determine what responsibilities are impacted by the change.<li data-bbox="1128 491 2332 678">▪ Determine an allocation of responsibilities to modules that places, as much as possible, responsibilities that will be changed (or impacted by the change) together in the same module, and places responsibilities that will be changed at different times in separate modules. |
| Coordination Model | <p>Determine which functionality or quality attribute can change at runtime and how this affects coordination; for example, will the information being communicated change at runtime, or will the communication protocol change at runtime? If so, ensure that such changes affect a small number set of modules.</p> <p>Determine which devices, protocols, and communication paths used for coordination are likely to change. For those devices, protocols, and communication paths, ensure that the impact of changes will be limited to a small set of modules.</p> <p>For those elements for which modifiability is a concern, use a coordination model that reduces coupling such as publish-subscribe, defers bindings such as enterprise service bus, or restricts dependencies such as broadcast.</p> |



Example: Design Checklist for Modifiability

| | |
|------------|---|
| Data Model | <p>Determine which changes (or categories of changes) to the data abstractions, their operations, or their properties are likely to occur. Also determine which changes or categories of changes to these data abstractions will involve their creation, initialization, persistence, manipulation, translation, or destruction.</p> <p>For each change or category of change, determine if the changes will be made by an end user, a system administrator, or a developer. For those changes to be made by an end user or system administrator, ensure that the necessary attributes are visible to that user and that the user has the correct privileges to modify the data, its operations, or its properties.</p> <p>For each potential change or category of change:</p> <ul style="list-style-type: none">▪ Determine which data abstractions would need to be added, modified, or deleted to make the change.▪ Determine whether there would be any changes to the creation, initialization, persistence, manipulation, translation, or destruction of these data abstractions.▪ Determine which other data abstractions are impacted by the change. For these additional data abstractions, determine whether the impact would be on the operations, their properties, their creation, initialization, persistence, manipulation, translation, or destruction.▪ Ensure an allocation of data abstractions that minimizes the number and severity of modifications to the abstractions by the potential changes. <p>Design your data model so that items allocated to each element of the data model are likely to change together.</p> |
|------------|---|



Example: Design Checklist for Modifiability

| Category | Checklist |
|--------------------------------------|--|
| Mapping among Architectural Elements | <p>Determine if it is desirable to change the way in which functionality is mapped to computational elements (e.g., processes, threads, processors) at runtime, compile time, design time, or build time.</p> <p>Determine the extent of modifications necessary to accommodate the addition, deletion, or modification of a function or a quality attribute. This might involve a determination of the following, for example:</p> <ul style="list-style-type: none"><li data-bbox="1128 476 1640 519">▪ Execution dependencies<li data-bbox="1128 519 1794 563">▪ Assignment of data to databases<li data-bbox="1128 563 2280 649">▪ Assignment of runtime elements to processes, threads, or processors <p>Ensure that such changes are performed with mechanisms that utilize deferred binding of mapping decisions.</p> |
| Resource Management | <p>Determine how the addition, deletion, or modification of a responsibility or quality attribute will affect resource usage. This involves, for example:</p> <ul style="list-style-type: none"><li data-bbox="1128 908 2280 995">▪ Determining what changes might introduce new resources or remove old ones or affect existing resource usage<li data-bbox="1128 995 2178 1038">▪ Determining what resource limits will change and how <p>Ensure that the resources after the modification are sufficient to meet the system requirements.</p> <p>Encapsulate all resource managers and ensure that the policies implemented by those resource managers are themselves encapsulated and bindings are deferred to the extent possible.</p> |



Example: Design Checklist for Modifiability

Binding Time

For each change or category of change:

- Determine the latest time at which the change will need to be made.
- Choose a defer-binding mechanism (see Section 7.2) that delivers the appropriate capability at the time chosen.
- Determine the cost of introducing the mechanism and the cost of making changes using the chosen mechanism. Use the equation on page 118 to assess your choice of mechanism.
- Do not introduce so many binding choices that change is impeded because the dependencies among the choices are complex and unknown.

Choice of Technology

Determine what modifications are made easier or harder by your technology choices.

- Will your technology choices help to make, test, and deploy modifications?
- How easy is it to modify your choice of technologies (in case some of these technologies change or become obsolete)?

Choose your technologies to support the most likely modifications. For example, an enterprise service bus makes it easier to change how elements are connected but may introduce vendor lock-in.



Software Architecture in Practice

see also:

- Bass et al.: Software Architecture in Practice, Ch. 15, 17



Structural Recommendations for Good Architectures

- Have well-defined, well-encapsulated modules.
- Use well-known architectural patterns and tactics to achieve quality attributes.
- Don't depend on a particular version of a particular product or tool, but structure the system so that swapping it out is straightforward and inexpensive.
- Separate modules that produce data from modules that consume data.
- Don't expect a one-to-one correspondence between design-time modules and their run-time instances.
- Ensure that any process can be (re)assigned to any processor, even at run time.
- Use the same small set of module interaction paradigms throughout the system.
- Ensure that areas of resource contention are small, and that clear conflict resolution mechanisms are specified for them.



Software Process Recommendations

Towards Good Architectures

- The architecture should be the responsibility of a single lead architect who has a strong connection to the development team, to ensure conceptual integrity.
- The architect should base the architecture on a prioritized list of well-specified quality attributes that inform the inevitable trade-offs.
- The architecture documentation should address the concerns (analysis, construction, maintenance, training) of the most important stakeholders.
- The architecture should be evaluated early and repeatedly for its ability to deliver the system's important quality attributes.
- The architecture should lend itself to incremental implementation
 - e.g. by building a skeletal system first in which just the communication paths are in place, and to which the functional components can be added incrementally.



Iterative-Incremental Development of Architecture

■ Inception

- Candidate architectures are considered.
- If architectural feasibility of key requirements is questionable, a light proof-of-concept architecture may be implemented to determine feasibility.

■ Elaboration

- Architectural drivers (i.e. major architectural risks) are identified and resolved.
- An executable architecture that can serve as the backbone of further implementation is built.

■ Construction

- Business components are built and integrated with the core architecture.
- No major changes of the architecture should be necessary.

■ Transition

- The final architecture is documented as a basis for future maintenance and evolution.



Up-Front Architecture Planning vs. Agility

- Pressure to deliver a working system with demonstrable features early can lead to a disregard of architectural concerns in an agile project
 - Each feature would be independently designed and implemented
 - Concerns cutting across more than one feature may be easily missed, resolved in redundant or incompatible ways
- In an architecture-centric project, up-front analysis of functional and quality requirements, and making suitable design decisions, likely leads to a stable architecture, but takes time in which the customer does not get tangible results
 - Ideally, agile projects will want to evolve the architecture as they go along, while traditional projects invest more time in up-front analysis and planning
 - Balance modifiability tactics with the agile YAGNI principle (“don’t do what you don’t have to do”)
- The whole point of choosing how much time to budget for architecture is to reduce financial / political / operational / reputational risk.

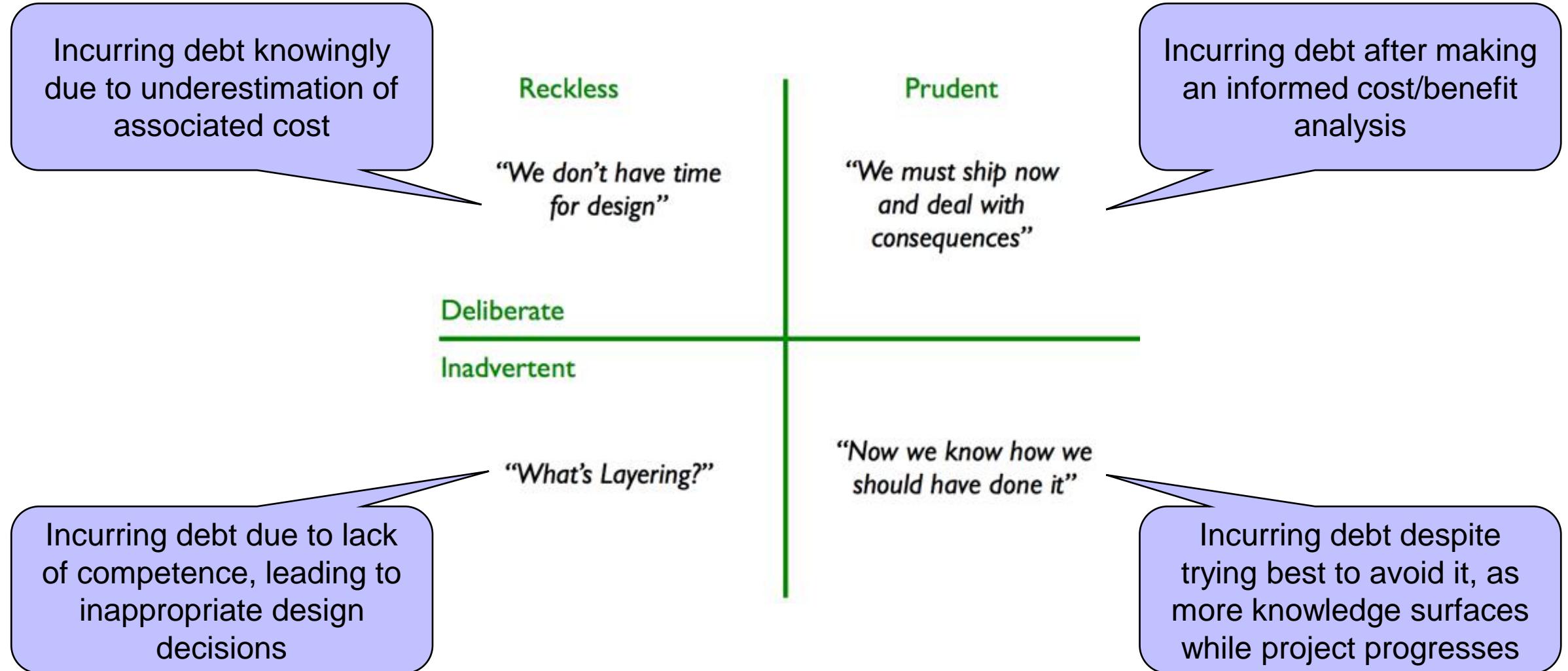
Up-Front Architecture Planning vs. Agility

- Project realities often force architects to deal with conflicting goals:
 - Pleasing the customer by enabling the team to deliver many requested features fast, vs.
 - Ensuring the system's longevity by designing for modifiability, extensibility, portability up front
 - Many requirements trade off against each other
 - e.g. security comes at expense of performance, modifiability comes at expense of time to market, availability and performance come at expense of modifiability and cost, etc.
- Strive to work from two perspectives simultaneously:
 - **Top-down view:** Design and analyze architectural structures that meet quality requirements and make appropriate trade-offs
 - **Bottom-up view:** Analyze wide array of implementation- and environment-specific constraints and find solutions to them
- The latter activities address questions (= project risks) that can better be answered experimentally than analytically
 - Experiments / prototypes to eliminate these risks are the focus of “spikes” in agile planning

Technical Debt

- Technical debt is most commonly incurred by quick-and-dirty implementations that incur penalties in the future, in terms of increased maintenance costs.
- The longer technical debt remains in the system, the higher the necessary maintenances costs due to the amount of code that needs to be refactored.
 - “As an evolving program is continually changed, its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it.” (Meir M. Lehman, 1980)
 - “Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite... The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt.” (Ward Cunningham, 1992)
 - “Developers who evolve such systems suffer [due to] never being able to write quality code because they are always trying to catch up. Users suffer the consequences of capricious complexity, delayed improvements, and insufficient incremental change.” (Grady Booch, 2014)

Causes of Technical Debt



Suggested Architecture Focus in Different Project Types

- Large and complex systems with relatively stable, well-understood requirements
 - Perform a large amount of architecture work up front
 - Low risk of changes, high pay-off in quality
- Big projects with vague or unstable requirements
 - Start by designing a complete candidate architecture (even if sketchy and omitting details)
 - Make sure it's enough to demonstrate end-to-end functionality and link major functionality
 - Be prepared to elaborate and change this as necessary
 - Necessity for changes should be determined through spikes and prototypes
 - Architecture will not be perfect, but help guide understanding, development
- Smaller, less critical projects with uncertain requirements
 - Get agreement on central patterns to be employed
 - But don't spend too much time on up-front architecture analysis and documentation

The Pivotal Role of Software Architecture

- Having a good software architecture is a key to a project's success because:
 - An architecture will inhibit or enable a system's driving quality attributes.
 - Architecture analysis enables early prediction of a system's qualities.
 - Architecture decisions allow you to reason about and manage system change and evolution.
 - A documented architecture enhances communication among stakeholders.
 - An architecture carries the earliest, most fundamental, hardest-to-change design decisions.
 - An architecture defines a set of constraints on subsequent implementation.
 - An architecture dictates the structure of an organization, or vice versa.
 - An architecture can provide the basis for evolutionary prototyping.
 - An architecture is the key artifact enabling reasoning about cost and schedule.
 - An architecture can be a transferable, reusable model at the heart of a product line.
 - An architecture focuses attention on assembly rather than just creation of components.
 - An architecture reduces design and system complexity by restricting design alternatives.
 - An architecture can be the starting point for introducing new team members to a project.





Hugbúnaðarverkefni 2 / Software Project 2

13. Final Exam Preparation

HBV601G – Spring 2020

Matthias Book



HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRVÍSINDASVIÐ
ÍÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD

In-Class Quiz #10 Solution: Software Architecture Summary

- A (a) **system architecture** can be considered from several perspectives:
 - (b) **Module structure**: Design-time view of divisions and relationships of constructed modules
 - (c) **Component-and-Connector structure**: Run-time view of interactions of module instances
 - (d) **Allocation structure**: Design- and run-time view of mapping arch. elements to each other
- A system's architecture is **established through a series of (e) design decisions.**
 - e.g. allocation of responsibilities, coordination model, data model, resource management, mapping of architectural elements, variations and binding times, technology choices
- Design decisions **influence (f) quality attributes** of the system.
 - e.g. availability, interoperability, modifiability, performance, security, testability, usability
- Decisions affect & depend on technical, project, business, professional (g) **contexts.**



Recap: Up-Front Architecture Planning vs. Agility

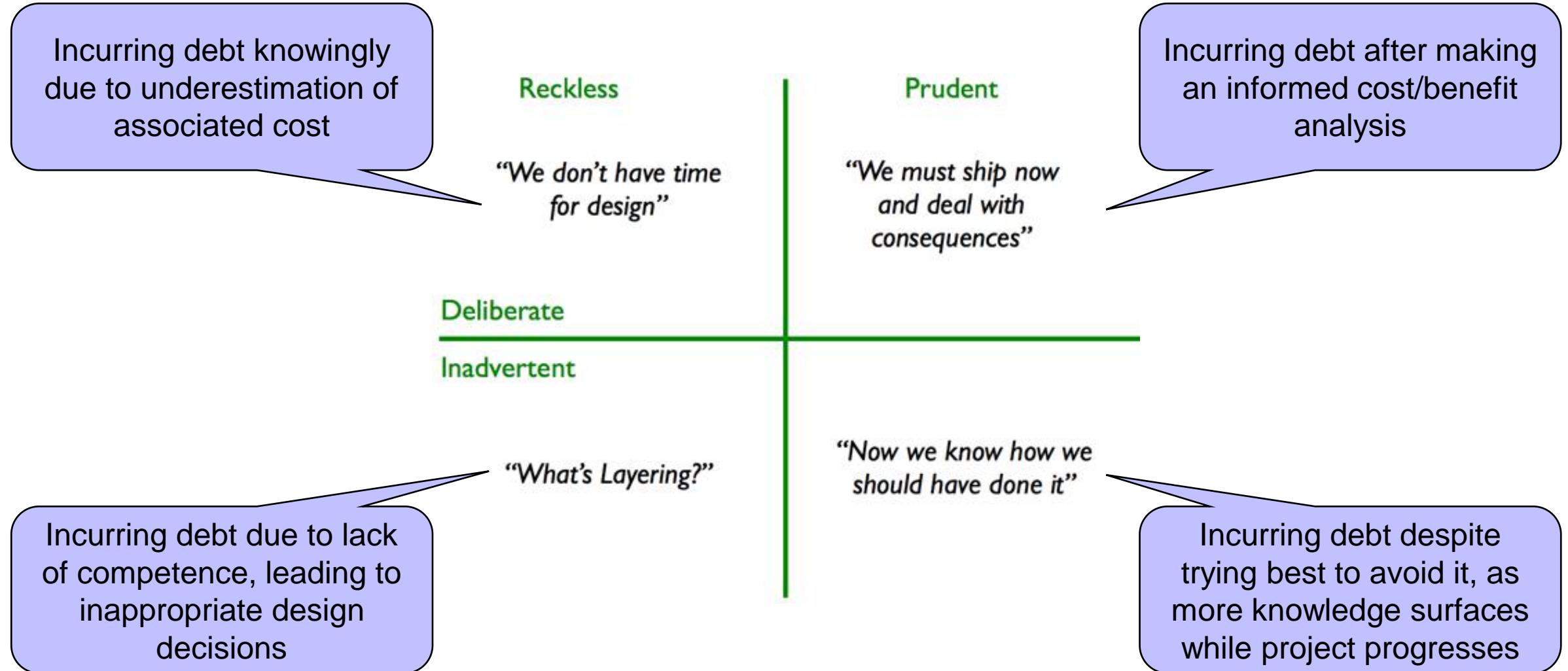
- Pressure to deliver a working system with demonstrable features early can lead to a disregard of architectural concerns in an agile project
 - Each feature would be independently designed and implemented
 - Concerns cutting across more than one feature may be easily missed, resolved in redundant or incompatible ways
- In an architecture-centric project, up-front analysis of functional and quality requirements, and making suitable design decisions, likely leads to a stable architecture, but takes time in which the customer does not get tangible results
 - Ideally, agile projects will want to evolve the architecture as they go along, while traditional projects invest more time in up-front analysis and planning
 - Balance modifiability tactics with the agile YAGNI principle (“don’t do what you don’t have to do”)
- The whole point of choosing how much time to budget for architecture is to reduce financial / political / operational / reputational risk.

Technical Debt

- Technical debt is most commonly incurred by quick-and-dirty implementations that incur penalties in the future, in terms of increased maintenance costs.
- The longer technical debt remains in the system, the higher the necessary maintenances costs due to the amount of code that needs to be refactored.
 - “As an evolving program is continually changed, its complexity, reflecting deteriorating structure, increases unless work is done to maintain or reduce it.” (Meir M. Lehman, 1980)
 - “Shipping first time code is like going into debt. A little debt speeds development so long as it is paid back promptly with a rewrite... The danger occurs when the debt is not repaid. Every minute spent on not-quite-right code counts as interest on that debt.” (Ward Cunningham, 1992)
 - “Developers who evolve such systems suffer [due to] never being able to write quality code because they are always trying to catch up. Users suffer the consequences of capricious complexity, delayed improvements, and insufficient incremental change.” (Grady Booch, 2014)



Causes of Technical Debt



Suggested Architecture Focus in Different Project Types

- Large and complex systems with relatively stable, well-understood requirements
 - Perform a large amount of architecture work up front
 - Low risk of changes, high pay-off in quality
- Big projects with vague or unstable requirements
 - Start by designing a complete candidate architecture (even if sketchy and omitting details)
 - Make sure it's enough to demonstrate end-to-end functionality and link major functionality
 - Be prepared to elaborate and change this as necessary
 - Necessity for changes should be determined through spikes and prototypes
 - Architecture will not be perfect, but help guide understanding, development
- Smaller, less critical projects with uncertain requirements
 - Get agreement on central patterns to be employed
 - But don't spend too much time on up-front architecture analysis and documentation



Summary: The Pivotal Role of Software Architecture

- Having a good software architecture early is key to a project's success because:
 - An architecture will inhibit or enable a system's driving quality attributes.
 - Architecture analysis enables early prediction of a system's qualities.
 - A documented architecture enhances communication among stakeholders.
 - An architecture carries the earliest, fundamental, hardest-to-change design decisions.
 - Architecture decisions allow you to reason about and manage system change and evolution.
 - An architecture defines a set of constraints on subsequent implementation.
 - An architecture dictates the structure of an organization, or vice versa.
 - An architecture can provide the basis for evolutionary prototyping.
 - An architecture is the key artifact enabling reasoning about cost and schedule.
 - An architecture can be a transferable, reusable model at the heart of a product line.
 - An architecture focuses attention on assembly rather than creation of components.
 - An architecture reduces design and system complexity by restricting design alternatives.
 - **An architecture can be the starting point for introducing team members to a project.**



Agile Practices

A Critical Review

Further reading:

- Bertrand Meyer: Agile! The Good, the Hype and the Ugly. Springer, 2014



Criticism of the Agile Manifesto

Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler

James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick

Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

© 2001, the above authors
this declaration may be freely copied in any form,
but only in its entirety through this notice.



HÁSKÓLI ÍSLANDS

Matthias Book:
Software Project 2

Testing...?

Principles behind the Agile Manifesto

We follow these principles:

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Overall impression:

unclear

partially unsubstantiated

hard to operationalize

Redundant

Redundant

Assertion

Redundant

Assertion

Assertion

Wrong

Assertion

Sources: agilemanifesto.org;
Bertrand Meyer: Agile Methods – The Good, the Hype and the Ugly (2014)

Agile Practices: “The Ugly”

- **Rejection of up-front tasks, particularly: no up-front requirements**
- **User stories as a replacement for abstract requirements**
- **Feature-based development and ignorance of dependencies**
- Tests as a replacement for specifications
- Test-driven development
- Embedded customer
- Coach and method keeper (e.g. Scrum Master) as a separate role
- Dismissal of traditional manager tasks
- **Dismissal of a priori architecture work**
- **Dismissal of a priori concern for extensibility**
- **Dismissal of a priori concern for reusability**
- Dismissal of auxiliary products and non-shippable artifacts



Agile Practices: “The Indifferent”

- Pair programming
- Open-space working arrangements
- **Self-organizing teams**
- **Maintaining a sustainable pace**
- Producing minimal functionality
- **Planning game, planning poker**
- **Cross-functional teams**



Agile Practices: “The Good”

- **Acceptance of change**
- **Frequent iterations**
- Emphasis on working code
- Tests as one of the key resources of the project
- Constant test regression analysis
- No branching
- Product (but not user stories!) **burndown chart**
- **Daily meeting**



Agile Practices: “The Brilliant”

- **Short iterations**
- **Closed-window rule**
- **Refactoring** (but not as a substitute for design)
- **Associating a test with every piece of functionality**
- **Continuous integration**



Suggestion: More Actionable Agile Principles

Organizational Principles

1. Put the customer at the center
2. Accept change
3. Let the team self-organize
4. Maintain a sustainable pace
5. Produce minimal software
 1. Produce minimal functionality
 2. Produce only the product requested
 3. Develop only code and tests

Technical Principles

8. Develop iteratively
 1. Produce frequent working iterations
 2. Freeze requirements during iterations
9. Treat tests as a key resource
 1. Do not start any new development until all tests pass
 2. Test first
10. Express requirements through scenarios

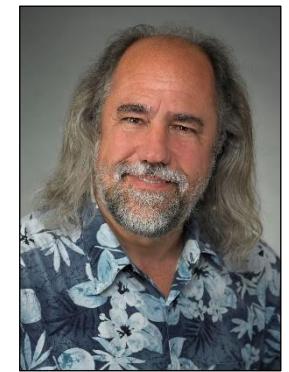


Food for Thought

- Meyer's preceding critical review is not established "textbook" knowledge, but a contribution to an ongoing academic and professional discussion.
- Some of his criticism will weigh differently depending on the type of project, the composition of the team, etc.
- It is provided here to spark and encourage your own critical thinking about the software engineering practices you have learned about.
- **How do you perceive the agile practices?**
 - Which ones would you adopt?
 - Which ones would you modify?
 - Why?



Grady Booch: The Future of Software Engineering (A Retrospective and Outlook on Our Discipline)



International Conference on Software Engineering 2015 Keynote

- No matter what future we may envision, it relies on software that has not yet been written. The nature of the systems we build continues to change, and as they weave themselves into our lives, we must attend not only to the technical elements of software development, we must also attend to human needs. This talk looks at the history of software engineering and offers some grand challenges for the future.
- **Grady Booch** is Chief Scientist for Software Engineering at IBM Research. Having originated the term and the practice of object-oriented design, he is best known for his work in advancing the fields of software engineering and software architecture. He is a co-author of the Unified Modeling Language (UML), a founding member of the Agile Alliance and the Hillside Group, has been awarded the Lovelace Medal and given the Turing Lecture for the BCS.

Grading Recap and Final Exam



HÁSKÓLI ÍSLANDS

Matthias Book: Software Project 2

Recap: Assignment 4: Schedule and Deliverables

- On **Thu 16 Apr**, demonstrate and explain your product to your classmates in a 15-minute-presentation (plus 5 minutes for questions and handover):
 1. **Product:** What does your product do? Demonstrate the key features of your system.
 2. **Architecture:** How does your product work? Explain architecture & key design decisions.
 3. **Process:** How did you build the product? Relate and interpret challenges you faced.
- On **Mon 20 Apr**, submit your **final product** in Uglá, incl.:
 - Complete source code and installation instructions
 - Slides of your final presentation
- Your product does not need to satisfy all the criteria you listed in your initial requirements document, but the key features should work.



Assignment 4: Final Presentation Schedule

- All teams will present their work in their TA's Zoom room, at the following times:

| Timeslot | Arnar's room | Kristján's room | Sigurður's room |
|-------------|--------------|-----------------|-----------------|
| 13:20-13:35 | 2 | 6 | 1 |
| 13:40-13:55 | 4 | 7 | 3 |
| 14:00-14:15 | 7 | 9 | 5 |
| 14:25-14:40 | 10 | 11 | 15 |
| 14:45-15:00 | 12 | 20 | 16 |
| 15:05-15:20 | 13 | 23 | |
| 15:30-15:45 | 21 | 24 | 22 |
| 15:50-16:05 | 27 | 26 | 25 |
| 16:10-16:25 | 29 | 30 | |

- All teams should be present at least during the hour of their presentation, i.e. 13:20-14:20, 14:25-15:25 or 15:30-16:30, so that every team has an audience.

Recap: Assignment 4: Grading

- Team grade refers to the visible parts of the presentation
 - i.e. information on slides, impression the product makes, etc. – Criteria:
 - Final product implements key features, and is running smoothly (75%)
 - Critical retrospective given of chosen process, architecture and technology (25%)
- Presenter grade refers to the audible parts of the presentation
 - Criteria: how well the presenter explains things, answers questions etc.
 - Presentation to tutor must be given by team member who has not presented any assignments yet (the other presentations may be given by other team members)
 - If all team members have gotten a presentation grade already, the presentation quality will be reflected in the team grade.



Recap: Overall Grading Scheme

Policy changes
for Spring 2020

- All contributing factors are graded on a scale of 0...11
 - Grading criteria for assignment deliverables
 - Individual presentation quality
 - Questions on in-class quizzes
 - Questions on final exam
- All factors are averaged using the published weights, without rounding
 - Exceptional performance (11) on some factors can outweigh lower performance elsewhere
- Resulting final grade is rounded to nearest half point
 - In case of a passed exam, final grade is capped at 10.0
 - ~~In case of a failed exam, final grade is capped at 4.5 (not applied this semester)~~
- Need a passing project grade to be allowed to final exam
 - If project grade is not sufficient, need to do a project again next year
- Weighted average of project and exam grade must be ≥ 5.0 to pass the course
 - If course is failed, exam can be re-taken during the resit period, and if course is failed again, the year after
 - No need (and not possible) to redo a passed project
 - Project grade remains valid for only one year though
- After receiving a passing course grade, you can apply to university to have it converted to “staðið”



Recap: Project Grading

- The project grade depends on the **deliverables** submitted and the **presentation** given for each assignment.
 - Grading criteria will be published together with assignment.
- All team members receive same grade for **deliverables** submitted for an assignment
 - Each assignment weighs **20%** of project grade
- Over the course of the semester, each team member must lead the **presentation** of at least one assignment to tutor
 - Focus: Don't just tell us what you did, but *why* you decided to do it this way.
 - The presenting team member receives an individual grade for their presentation ("5th assignment", weighs **20%** of project grade)
 - If someone gives several presentations, the best presentation grade counts
- The resulting project grade weighs 30-70% of the final course grade.



Recap: Grade Weights

Policy changes
for Spring 2020

- Peer assessment of teammates' contributions to project
 - At the end of the semester, all team members assess how much each of their teammates contributed to the project **in the time before the assembly ban** (i.e. before **16 March**).
 - Contribution votes are normalized to obtain each team member's contribution factor.
- Depending on team members' project contributions, the weight of their project and final exam grades will be individually adjusted between 30% and 70%:
 - Below-average contribution → lower weight of project grade, higher weight of exam grade
 - Average contribution → equal weight of project and exam grade (50:50)
 - Above-average contribution → higher weight of project grade, lower weight of exam grade
- In rare cases where the above rules would punish someone who contributed above average, or reward someone who contributed below average, the traditional 50:50 weight distribution is used instead.

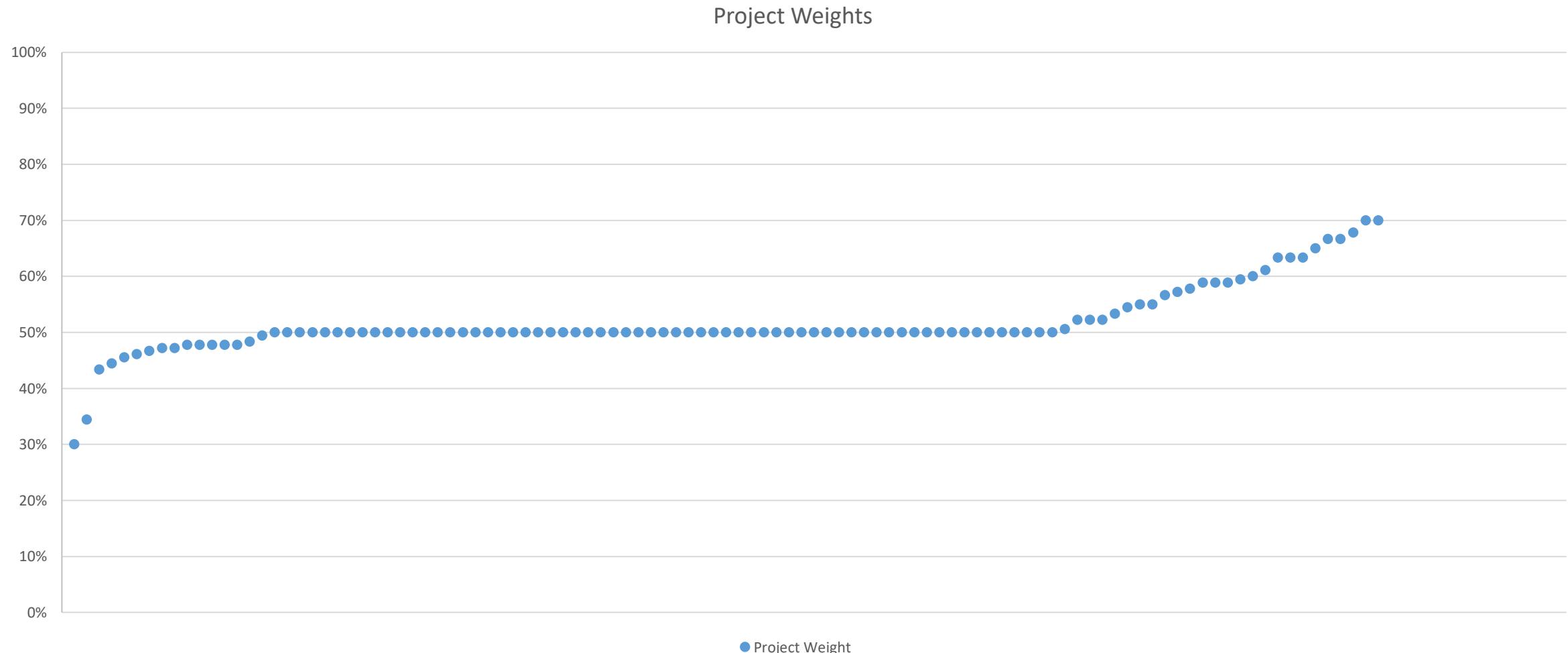


Peer Assessment

- Each team member can assess the contribution of each of their teammates:
 - +++ : contributed **much more** than others
 - ++ : contributed **more** than others
 - + : contributed **slightly more** than others
 - o : contributed **same** as others
 - - : contributed **slightly less** than others
 - -- : contributed **less** than others
 - --- : contributed **much less** than others
- Assessments for each team member will be averaged, and the assessments will be normalized across all team members
 - if you and your team mates all rate each other as “+++”, your normalized contribution will be “o”
- Ratings should reflect a fair assessment of your own and your teammates’ contribution.
 - Ultimate decision of how assessments are counted remains with tutors.
- Submit your ratings under the “Peer Assessment” assignment in Uglá by **Mon 20 Apr.**



Example: A Past Course's Project Weights based on Peer Assessment



Recap: Final Exam

Policy changes
for Spring 2020

- **Date & Time:** Mon 27 Apr, 09:00-13:30
 - Exam designed for 3 hours
 - but everyone gets 4.5 hours exam time
- **Support:** I'll answer questions anytime
 - <https://eu01web.zoom.us/j/825013950>
- **Focus:** Understanding of software engineering concepts and methods
- **Scope:** Lecture slides (i.e. contents of Námsefni folder)
 - Note: The soundtrack is relevant!
- **Style:** Online take-home exam
 - Type your answers into Gradescope
 - Mark exam only with your exam number, *not* your name
- **Weight:** 30-70% of course grade

- **Tools:**
 - Open-book exam
 - You may consult your lecture slides, notes, dictionary etc.
 - Still highly recommended to create a cheat sheet for training purposes!
 - No collaboration allowed (no IM, forums...)
 - I reserve the right to invite you to an oral exam to determine your grade, if I have the impression your answers are not your own.
- **Questions:**
 - 10 questions with same weight
 - Two worst questions discarded
- **Answers:**
 - in English, **in your own words(!)**
 - short paragraphs of whole sentences



In Exchange for this Flexibility...

Policy changes
for Spring 2020

- **No collaboration on exams**, please – you're on your honor!
- I will place stronger emphasis on answers being **in your own words**.
 - Answers that mirror phrases on my slides will receive reduced points.
 - Answers that mirror other students' answers will receive zero points.
- If I have the strong impression that parts of an exam are not your own work, I reserve the right to invite you to an oral exam to determine the exam grade.
- **Don't stress out** about these rules – if you're working on your own, you're fine.



Recap: Optional In-Class Quizzes

Policy changes
for Spring 2020

- To encourage class attendance: Small quizzes in most lectures
 - Solved and handed in during class
 - Graded on usual scale, with 0 for any unanswered questions, 5...11 for answered questions
 - Three worst quiz grades (e.g. from skipped classes) will be ignored
 - Grades of all in-class quizzes will be averaged into one final **quiz grade**
 - Quiz grade can improve your final exam grade:
 - 8 out of 10 final exam questions add up to 100%
 - Quiz grade will be counted as an optional additional final exam question worth 7.5%
- If you didn't participate in any quizzes, you can still get top marks on the exam
- If you did participate in quizzes, the quiz grade can improve your exam grade by up to 11 on a 7.5% question, and thereby make up for deficiencies elsewhere



Online Exam Trial Run

Policy changes
for Spring 2020

- There'll be an opportunity to **try out the online exam tool and process** in the lecture timeslot after Easter (Thu 16 Apr 08:20-09:50)
 - Using the electronic exam system
 - Uploading solutions
 - Calling for teacher's assistance
 - etc.
- More details on this to come next week



Sample Exam Questions

- The following review questions are representative of the types of questions that could be on an exam.
- The answers to some of the following review questions can be found immediately on the lecture slides.
 - Caution: An open-book exam will obviously contain very few of these!
- Most questions require more individual reasoning and/or application of learned knowledge.
 - Be precise, justify your answers.

■ **Read exam questions carefully**

- Are you asked to explain or give an example or both?
- Are you asked to argue for or discuss an issue?
- Are you asked to explain or discuss an issue, or both?
- How many concepts are you asked to consider?

■ **Answer precisely**

- i.e. brief, focused, comprehensive



Types of Questions

(Answer in Your Own Words!)

- “**Explain...**”
 - Give an explanation of what something is or how something works in general.
 - Note: A concrete example is not a substitute for a general explanation.
- “**Argue...**”
 - Make up your mind about a certain issue.
 - Present arguments for your opinion on the issue.
- “**Discuss...**”
 - Consider both sides of an issue.
 - Present arguments for both sides.
 - You can state your own opinion, but should present counter-arguments as well.
- “**Suggest...**”
 - Come up with a solution for an issue.
 - Briefly explain what should be done.
- “**Give an example...**”
 - Give a brief example that illustrates the considered concept.
 - When giving examples for several distinct concepts, make sure the examples highlight the distinguishing characteristics.
- “**Point out issues...**”
 - Examine a given artifact, and explain what is wrong with it.
 - Don’t just say *what* is wrong, but *why* it is.
- “**Draw...**”
 - ~~Draw a UML diagram reflecting the relevant aspects of the given scenario.~~
 - ~~Pay attention to proper syntax!~~
- “**Implement...**”
 - ~~Write/modify brief segments of Java code.~~
 - ~~Pay attention to proper syntax!~~



Sample Questions: Software Requirements

- Argue why neither “Let’s just start coding and see what the customer says” nor “Let’s get the complete and precise requirements from the customer first” are effective approaches to requirements engineering.
- Explain the difference between needs, features and requirements.
- Initial versions of user stories that a team comes up with are often too broad. Explain two patterns for refining user stories by splitting them into more stories.
- Argue how much emphasis should be placed on requirements engineering in plan-driven projects and in agile projects.
- Discuss whether a traditional product manager could fill the role of an agile product owner.



Sample Questions: Software Estimation

- Argue why single-point estimates can be misunderstood, and how estimates should be phrased instead.
- Explain how much divergence between an estimate and a target can typically be coped with through project management, and argue what should be done when the divergence is bigger.
- For the following scenarios, calculate the projected effort, or argue why a projection is not possible: [...]
- Assume that for a certain feature, an expert came up with a best-case estimate of ... person-days (PD), a most-likely-case estimate of ... PD and a worst-case estimate of ... PD. Calculate the expected-case estimate according to the basic PERT formula.



Sample Questions: Software Estimation

- Argue whether it is a good sign if about half of your estimates are above and about half are below the actual values.
- Explain the differences between Wideband Delphi and Planning Poker.
- Argue whether you would use the T-shirt sizing method for sprint planning.



Sample Questions: Android Development

- Explain the relationship between an activity and a layout.
- Explain why anonymous inner classes are frequently used in Android development.
- Explain how you can avoid losing an activity's state when the device is rotated.
- Explain the difference between explicit and implicit intents.



Sample Questions: Android Development

- Explain for what kinds of data you would use a) a **SharedPreferences** file, b) a binary file, and c) an SQLite database in your app's internal storage directory.
- Explain the difference between internal and external file storage.
- Explain the role of **@DAO** objects when using the Room persistence framework.
- Explain the difference between **AsyncTasks** and **HandlerThreads**.



Sample Questions: Software Architecture

- Explain the difference between the module structure and the allocation structure of a software architecture.
- Name two types of design decisions shaping the module structure of a software architecture, and give an example for each type of decision.
- Assume you want to improve the performance of a software system. Give two examples of design considerations that would shape your technology choices.



Sample Questions: Software Engineering Practices

- Consider Meyer's classification of agile practices into “the ugly”, “the indifferent”, “the good” and “the brilliant”. Pick one practice whose classification you agree with and one that you disagree with, and argue why you agree or disagree with them.
- Pick a software engineering practice that you learned about in HBV401G, HBV501G or HBV601G. Argue under which conditions you believe it would work well, and suggest how it could be modified for different conditions.



Wrap-up



HÁSKÓLI ÍSLANDS

Matthias Book: Software Project 2

Special Thanks

Arnar Þór Sigurðsson
Kristján Pétur Þórarinsson
Sigurður Gauti Samúelsson



Kennslukönnun

Evaluate this course on Uglar!



Long-Term Feedback

- Over the course of your future academic and professional career, you'll be exposed to challenges
 - that these courses prepared you for
 - that these courses didn't prepare you for
- I'd appreciate if you could let me hear about these in 1, 2, 3... years so we can keep tailoring the curriculum better to actual demands.
 - What could you use?
 - What couldn't you use?
 - What do you wish you had been taught?
 - What do you wish you had taken more seriously?
- Contact me at **book@hi.is** – I look forward to keep hearing from you!

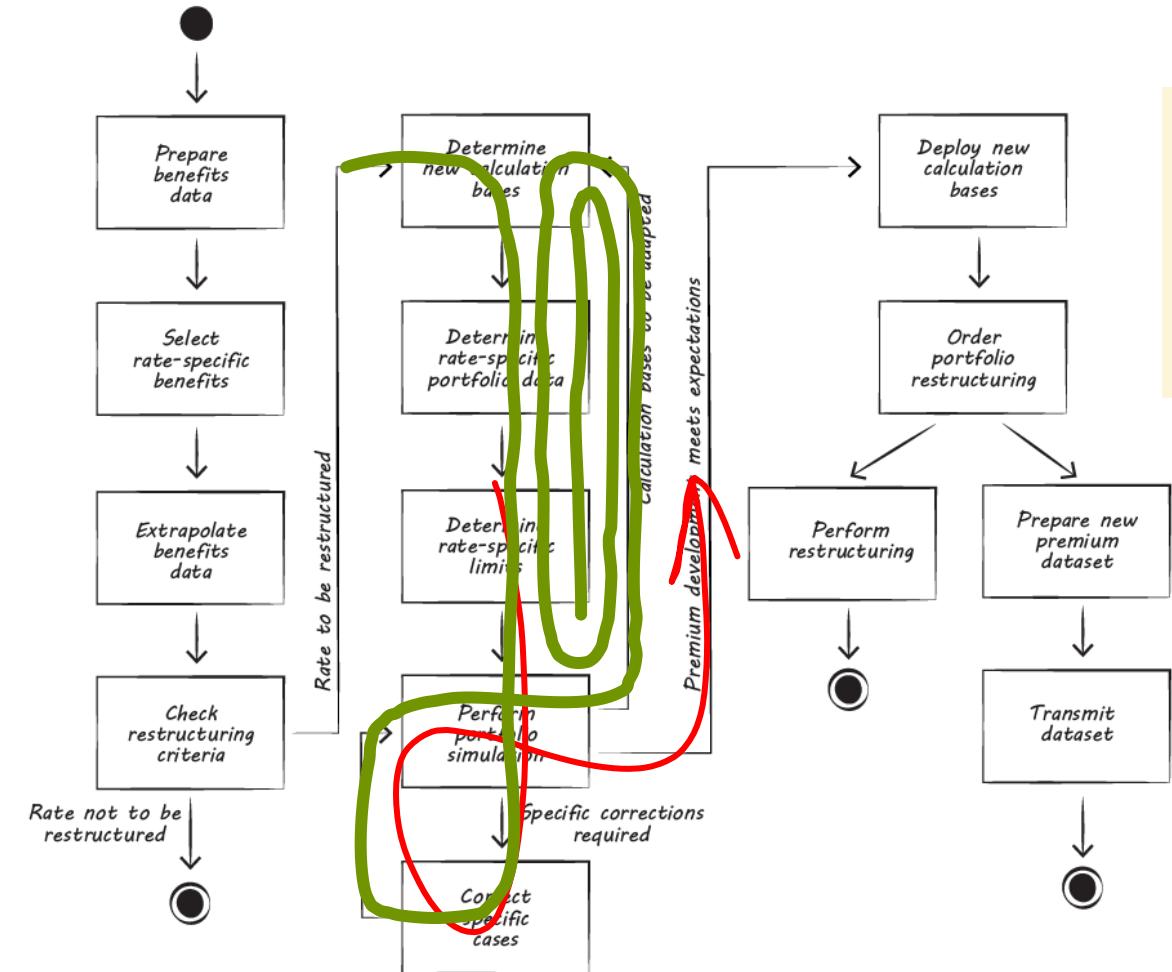


Teaching Assistants Wanted

- I am looking for TAs for next semester's course:
 - **HBV501G Software Project 1**
 - If you are interested or know someone who is, please contact me at **book@hi.is**.
- **Tasks**
 - Advising student teams
 - Scoring assignments
 - **Opportunities**
 - Help to shape the course
 - Brush up your CV
 - **Requirements**
 - Successful completion of respective course
 - Ability to advise and evaluate student teams



Recap: Sketch-based Software Engineering



Goal: Enable developers to use sketching

- not just to help them think about software engineering activities
- but as an interaction modality to perform and complete activities directly

Code snippets illustrating sketch-based test case specification and code merging:

```

<constraint
    firstAttribute="trailing"
    secondItem="LFs-rG-Cv6"
    secondAttribute="trailing"
    id="w6V-ZV-c3M"/>
  </constraints>
</view>
<connections>
  <outlet property="displaySubtitle"
    destination="PPq-nc-dk0" id="p52-Mf1lw"/>
  <outlet property="bottomView"
    destination="7Wa-hI-bwz" id="wcM-eJ-2e5"/>
  <outlet property="finishSuiDuButton"
    destination="dwz-XU-8U4" id="ttA-QnVFO"/>
  <outlet property="mainView"
    destination="LFs-rG-Cv6" id="OZC-zg-yYF"/>
  <outlet property="numberLabel"
    destination="zog-c3-fbX" id="wJq-Xf-MiY"/>
  <outlet property="weiTiaoView"
    destination="58Q-XU-7cJ" id="fre-Wb-wsR"/>
</connections>
</viewController>
<placeholder placeholderIdentifier=
  "IBFirstResponder" id="bFU-wk-q91"
  userLabel="First Responder"
  sceneMemberID="firstResponder"/>

```

Example: Sketch-based Test Case Specification



Example: Sketch-based Code Merging

60 ECTS HBV/TÖL MSc Project Topics in Sketch-based Software Engineering

- *12-month paid MSc student jobs working on the following research goals:*
- **Turning software artefacts into base layers** that can be sketched on (models, source code, running applications, etc.)
 - Semantic challenge: To correctly interpret user intentions, tools need to understand semantics of the user's sketches and elements of the base layer (i.e. source code)
 - Technical challenge: Reinvention of current stand-alone sketching tools' architecture, as sketching will turn into one interaction layer among others in general IDEs
- **Evaluating** how the sketch-based interaction can be integrated seamlessly with the developers' thoughts and ways of approaching software engineering tasks
- *If you are interested, please send your CV and grade transcript to **book@hi.is**.*

Keep in Touch!

- For feedback, questions, projects, theses, events, guest talks, collaboration...



book@hi.is



<http://is.linkedin.com/in/matthiasbook>



<http://www.facebook.com/matthiasbook>



<http://www.twitter.com/matthiasbook>

Takk fyrir og gangi þér vel!

book@hi.is



HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRVÍSINDASVIÐ

IÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD





Hugbúnaðarverkefni 2 / Software Project 2

14. Trial Exam

HBV601G – Spring 2020

Matthias Book



HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRVÍSINDASVIÐ
ÍÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD

Recap: Final Exam

Policy changes
for Spring 2020

- **Date & Time:** Mon 27 Apr, 09:00-13:30
 - Exam designed for 3 hours
 - but everyone gets 4.5 hours exam time
- **Support:** I'll explain questions via Zoom
 - anytime during the exam
- **Focus:** Understanding of software engineering concepts and methods
- **Scope:** Lecture slides (i.e. contents of Námsefni folder)
 - Note: The soundtrack is relevant!
- **Style:** Online take-home exam
 - Type your answers into Gradescope
 - Mark exam only with your exam number, *not* your name
- **Weight:** 30-70% of course grade

- **Tools:**
 - Open-book exam
 - You may consult your lecture slides, notes, dictionary etc.
 - Still highly recommended to create a cheat sheet for training purposes!
 - No collaboration allowed (no IM, forums...)
 - I reserve the right to invite you to an oral exam to determine your grade, if I have the impression your answers are not your own.
- **Questions:**
 - 10 questions with same weight
 - Two worst questions discarded
- **Answers:**
 - in English, **in your own words(!)**
 - short paragraphs of whole sentences



Online Exam Access

- I've registered you in a Gradescope course for HBV601G
 - Please let me know if you did not receive an invitation in your hi.is mail account!
- You'll find a Trial Exam on Gradescope as of this morning.
 - Submission closes on April 26, so you can try it out over the next few days.
- You'll find the actual Final Exam on Gradescope on April 27 at 08:00 already.
 - You can start earlier than 09:00, to give you some flexibility in case you have another exam in the afternoon.
 - Submission closes 4.5 hours after opening the exam, but latest at 13:30 on exam day.
 - Start at 09:00 at the latest to have the full exam time available.
 - You can't pause the timer, but the exam is designed for 3 hours, so you have plenty of time.



Ungraded Trial Exam

- The trial exam is an online recreation of last year's HBV601G exam, to give you a realistic idea of the exam's scope and look & feel.
- Try answering a few questions to see how things work.
- This is also helpful for me to see how your answers look on the back-end.
- Note that the trial exam will not be graded or evaluated in any way.



Opening the Exam



Your Courses

Your Courses

Welcome to Gradescope! Click on one of your courses to the right, or on the Account menu below.

Spring 2020

HBV601G
Software Project 2

1 assignment



Account



HBV601G

Software Project 2



Regrade Requests

INSTRUCTOR



Matthias



HBV601G | Spring 2020

NAME

Final Exam *DEMO* (not graded for course credit)

STATUS

No Submission

RELEASED

APR 16

DUE (UTC)

3 hours, 14 minutes left

APR 16 AT 1:30PM

Final Exam *DEMO* (not graded for course credit)

RELEASE DATE (UTC)

Apr 16 at 8:00AM

DU^E DATE (UTC)

Apr 16 at 1:30PM

MAXIMUM TIME PERMITTED

270 minutes

This is a timed assignment. You can start the assignment any time until the due date and have at most 270 minutes to finish. You should start by Apr 16 at 9:00AM (UTC) to have the full time available to you.

Submissions will not be accepted after the time expires or after the due date.

During the submission period, you can submit any number of times. You will be graded based on your final submission.

Start Assignment



HÁSKÓLI ÍSLANDS

Matthias Book:

Entering Answers

Count includes
subquestions

Time remaining
(click to hide/show)

- Enter the answers to the questions as plain text right in the web browser.
- Gradescope automatically saves your answers as you enter them.

The screenshot shows the Gradescope interface for a "Final Exam *DEMO*". At the top, it displays "3/17 Questions Answered" and "Saved at 8:43 AM". On the right, a timer shows "TIME REMAINING 3 hrs 59 mlns". A callout bubble points to the top right corner with the text "Count includes subquestions". The main content area contains two questions:

Q1 Software Requirements I
10 Points
Explain the differences between epics, features and requirements with regard to their scope and timeframe.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. In iaculis lorem ut eros tincidunt elementum. Duis at ante vel est ultricies mollis eget ac ex. Duis nibh nunc, hendrerit vel suscipit quis, tempus ac dolor. Aliquam ut pulvinar ipsum. Proin varius mattis mollis. Mauris in mollis metus, malesuada bibendum dui. Integer laoreet ante vitae lacus dictum ultrices.

Q2 Software Requirements II
10 Points
Argue how the role of a product owner differs from the role of a product manager.
Enter your answer here



Checking your Submission

- You can view what Gradescope has saved by clicking “View Your Submission” at the bottom of the exam page.

The screenshot shows the Gradescope interface. At the top, it displays "3/17 Questions Answered" and "Saved at 8:43 AM". On the right, it shows "TIME REMAINING" with "3 hrs 51 mlns". The main content area is titled "Q12 Appendix" with "0 Points". Below the title, there's a section titled "Types of Questions (Answer in Your Own Words!)" which lists several types of questions with their respective guidelines:

- "Explain..."
 - Give an explanation of what something is or how something works in general.
 - Note: A concrete example is not a substitute for a general explanation.
- "Argue..."
 - Make up your mind about a certain issue.
 - Present arguments for your opinion on the issue.
- "Discuss..."
 - Consider both sides of an issue.
 - Present arguments for both sides.
 - You can state your own opinion, but should present counter-arguments as well.
- "Suggest..."
 - Come up with a solution for an issue.
 - Briefly explain what should be done.
- "Give an example..."
 - Give a brief example that illustrates the considered concept.
 - When giving examples for several distinct concepts, make sure the examples highlight the distinguishing characteristics.
- "Point out issues..."
 - Examine a given artifact, and explain what is wrong with it.
 - Don't just say *what* is wrong, but *why* it is.
- "Draw..."
 - Draw a UML diagram reflecting the relevant aspects of the given scenario.
 - Pay attention to proper syntax!
- "Implement..."
 - Write/modify brief segments of Java code.
 - Pay attention to proper syntax!

At the bottom of the page, there are two buttons: "View Your Submission" and a user icon.



Read-Only View

The “points” below are NOT intermediate assessments, but just the percentage weights of the questions.

- This shows your answers as I'll see them during grading.
- Click on “Resubmit” in the bottom right corner to continue working on the exam.
 - This button does not “resubmit” anything, it just takes you back to the editing screen.

The screenshot shows a digital exam interface. On the left, there are two questions:

Q1 Software Requirements I
10 Points
Explain the differences between epics, features and requirements with regard to their scope and timeframe.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. In iaculis lorem ut eros tincidunt elementum. Duis at ante vel est ultricies mollis eget ac ex. Duis nibh nunc, hendrerit vel suscipit quis, tempus ac dolor. Aliquam ut pulvinar ipsum. Proin varius mattis mollis. Mauris in mollis metus, malesuada bibendum dui. Integer laoreet ante vitae lacus dictum ultrices.

Q2 Software Requirements II
10 Points
Argue how the role of a product owner differs from the role of a product manager.

At the bottom left, there is a button labeled "Select a question." with a user icon. At the bottom right, there are two buttons: "Submission History" and a red-circled "Resubmit" button with an upward arrow icon.

On the right side, there is a sidebar with the following information:

Final Exam *DEMO* • UNGRADED (not graded for course credit)

STUDENT
Matthias Test

TOTAL POINTS
- / 100 pts

QUESTION 1
Software Requirements I 10 pts

QUESTION 2
Software Requirements II 10 pts

QUESTION 3
Software Estimation I 10 pts

QUESTION 4
Software Estimation II 10 pts

4.1 (no title) 3.334 pts

4.2 (no title) 3.333 pts

4.3 (no title) 3.333 pts

QUESTION 5
Android Development I 10 pts

QUESTION 6

Resuming an Exam

Note: Interrupting the exam (e.g. by closing the browser) does NOT stop the timer!

CAUTION: Gradescope bug: The timer on this screen may be incorrect! Only the timer shown on the exam screen is accurate!

The screenshot shows the Gradescope interface. On the left, there's a sidebar with 'Your Courses' and 'Account' sections. The main area shows 'Your Courses' for 'Spring 2020' with 'HBV601G Software Project 2' selected (circled in red). The right side shows the course details for 'HBV601G Spring 2020'. It lists an assignment named 'Final Exam *DEMO*' (circled in red), which is 'Submitted' and due 'APR 16 AT 1:30PM' with '4 hours, 27 minutes left'. Below this, the exam itself is displayed with questions like 'Q1 Software Requirements I' and a large text box containing placeholder text. At the bottom, there's a 'Select a question.' button and a 'Resubmit' button (circled in red).

Your Courses

Welcome to Gradescope! Click on one of your courses to the right, or on the Account menu below.

HBV601G
Software Project 2
1 assignment

INSTRUCTOR

HBV601G | Spring 2020

NAME STATUS RELEASE DUE (UTC)

Final Exam *DEMO* (not graded for course credit) Submitted APR 16 APR 16 AT 1:30PM 4 hours, 27 minutes left

Q1 Software Requirements I
10 Points
Explain the differences between epics, features and requirements with regard to their scope and timeframe.
Lorem ipsum dolor sit amet, consectetur adipiscing elit. In iaculis lorem ut eros tincidunt elementum. Duis at ante vel est ultricies mollis eget ac ex. Duis nibh nunc, hendrerit vel suscipit quis, tempus ac dolor. Aliquam ut pulvinar ipsum. Proin varius mattis mollis. Mauris in mollis metus, malesuada bibendum dui. Integer laoreet ante vitae lacus dictum ultrices.

STUDENT
Matthias Test

TOTAL POINTS
- / 100 pts

QUESTION 1
Software Requirements I 10 pts

QUESTION 2
Software Requirements II 10 pts

QUESTION 3
Software Estimation I 10 pts

Select a question.

Submission History Resubmit



Exam Submission

- To formally confirm that you would like to submit the exam, please check the box next to the declaration in Question 11.
 - This does not need to be your last activity
 - Just do it anytime before the timer expires

The screenshot shows a digital exam interface. At the top, it displays "3/17 Questions Answered" and "Saved at 8:43 AM". On the right, it shows "TIME REMAINING" with "3 hrs 55 mlns". The main area features a question titled "Q11 Exam Submission" worth "0 Points". A note states: "! NOTE: THIS DEMO EXAM IS FOR TOOL PRACTICE ONLY, BUT NOT FOR COURSE CREDIT !". Below this, a message encourages frequent submissions: "You are encouraged to submit individual answers frequently as you enter and revise them, in order to prevent accidental loss of your work. However, YOUR EXAM WILL ONLY COUNT AS FORMALLY SUBMITTED IF YOU CHECK THE BOX BELOW AND SUBMIT THIS ANSWER." A checkbox is present with the text: "I hereby formally submit this exam. I confirm that this work is my own and I have not received any aid from others. I understand that a violation of the rules of the University of Iceland may lead to heavy disciplinary sanctions, ranging from impact on the final grade of the course, to expulsion from the University, on a temporary or permanent basis." The checkbox is circled in red.



Answer Backup

- To be on the safe side and not depend 100% on Gradescope, I recommend copying & pasting your final answers to a local text document occasionally.
- Upload that document as a PDF to Uglá in the “Exam Backup” assignment as soon as you have completed the exam.
 - I won’t look at this document (and it won’t have any formal meaning) unless there is a technical issue with Gradescope that requires me to fall back on this.
 - I can see at what time you opened the exam in Gradescope, and at what time you submitted the backup document in Uglá, so I can still check compliance with the 4.5 hour time limit.



Live Support

- If you have questions, you can find me at <https://eu01web.zoom.us/j/825013950>
 - until 10:00 today
 - from 08:00 to 13:30 on April 27
- You'll first enter a waiting room,
to ensure I'm talking to only one student at a time.
- I'll give the same level of support I would normally give during an exam,
i.e. not giving answers away, but explaining the questions.



Takk fyrir og gangi þér vel!

book@hi.is



HÁSKÓLI ÍSLANDS
VERKFRÆÐI- OG NÁTTÚRVÍSINDASVIÐ

IÐNAÐARVERKFRÆÐI-, VÉLAVERKFRÆÐI-
OG TÖLVUNARFRÆÐIDEILD

