**1. Activity** - це один із основних програмних компонентів в архітектурі додатків для операційної системи Android. Activity відповідає за користувацький інтерфейс додатку і взаємодію з користувачем. Цей компонент відображає користувацький інтерфейс на екрані пристрою і обробляє події, що виникають від користувача, такі як натискання на кнопки, дотики до екрану, обробка жестів тощо.

Основне призначення Activity в Android:

* Відображення інтерфейсу: Activity відповідає за відображення користувацького інтерфейсу додатку на екрані пристрою. Він може містити елементи, такі як кнопки, тексти, зображення, поля для введення даних і багато іншого.
* Обробка подій: Activity реагує на події, що виникають під час взаємодії користувача з додатком. Наприклад, він обробляє натискання кнопок, введення тексту користувачем, дотики до екрану та інші події.
* Управління життєвим циклом: Activity має свій власний життєвий цикл, що включає стани, такі як створення, запуск, призупинення, відновлення та знищення. Розробник може визначити, як Activity повинен вести себе під час переходу між цими станами.
* Міжкомпонентна взаємодія: Activity може взаємодіяти з іншими програмними компонентами, такими як інші Activity, Service, Broadcast Receiver і Content Provider. Це дозволяє створювати складні додатки, які взаємодіють з різними частинами системи та іншими додатками.

Activity є ключовим компонентом для створення інтерактивних додатків для Android, і він може бути розширений та налаштований для вирішення різних завдань, пов'язаних з відображенням інтерфейсу та взаємодією з користувачем.

**2. Об'єкт Intent і поняття Intent Filter** використовуються в архітектурі операційної системи Android для передачі повідомлень між компонентами додатків і взаємодії між ними. Давайте розглянемо їх ближче:

Intent (намір):

Intent - це об'єкт, який використовується для відправки запиту на виконання дії або передачі даних між різними компонентами додатків в Android, такими як Activity, Service, Broadcast Receiver тощо.

Intent може містити інформацію про дію, яку потрібно виконати, і дані, які пов'язані з цією дією. Наприклад, ви можете використовувати Intent для запуску нової Activity, передачі даних з однієї Activity в іншу, або виклику служби для виконання певних завдань.

Intent Filter (фільтр намірів):

Intent Filter - це опис, який компонент (зазвичай, Activity або Broadcast Receiver) надає системі для вказівки, які Intent він може обробляти.

Intent Filter містить інформацію про дії та категорії, які цей компонент готовий обробляти. Це допомагає системі знаходити підходящий компонент для обробки Intent, коли відбувається відправка такого Intent.

Наприклад, якщо ви маєте Activity, яка може відкривати фотографії, її Intent Filter може містити дію "VIEW" та категорію "DEFAULT", що означає, що ця Activity може бути викликана для перегляду фотографій.

За допомогою Intent і Intent Filter розробники можуть створювати взаємодію між різними компонентами додатків і навіть між різними додатками. Це дозволяє створювати багатофункціональні додатки, які можуть взаємодіяти між собою та іншими додатками в системі Android.

**3. Поняття фрагментів. Призначення та можливості фрагментів. Переваги використання фрагментів.**

Фрагменти (Fragments) - це один з ключових концепцій в розробці додатків для операційної системи Android. Фрагменти є частинами користувацького інтерфейсу, які можна вбудовувати в активності (Activities) додатку. Кожен фрагмент представляє собою окрему частину користувацького інтерфейсу та має свою логіку.

**Призначення та можливості фрагментів:**

Поділ користувацького інтерфейсу: Фрагменти дозволяють поділити користувацький інтерфейс на менші, незалежні частини, що спрощує розробку і підтримку додатку.

Багатовіконний інтерфейс: Фрагменти дозволяють створювати багатовіконний інтерфейс, де одна активність може містити кілька фрагментів, що можуть бути додані, замінені або видалені динамічно.

Відновлення стану: Фрагменти допомагають зберегти і відновити стан активності при зміні орієнтації пристрою або інших подібних подіях.

Взаємодія між фрагментами: Фрагменти можуть легко взаємодіяти між собою та з активністю, що їх містить. Це дозволяє створювати додатки з складною взаємодією між різними частинами користувацького інтерфейсу.

Переносність: Фрагменти дозволяють писати код, який може бути використаний в різних активностях, що сприяє переносність і повторному використанню коду.

**Переваги використання фрагментів:**

Підтримка різних розмірів екрану: Фрагменти допомагають легше адаптувати додаток до різних розмірів екрану, включаючи планшети і смартфони.

Багатозадачність: За допомогою фрагментів можна створити багатозадачний додаток, де різні фрагменти можуть працювати паралельно.

Код з більшим рівнем абстракції: Фрагменти дозволяють розділити код на менші та більш розумні частини, що полегшує розробку та підтримку додатку.

Відновлення стану: Фрагменти автоматично допомагають у відновленні стану додатку при зміні орієнтації пристрою або інших подібних ситуаціях.

Фрагменти дозволяють розробникам створювати більш гнучкі та потужні додатки для Android, які легше адаптувати до різних сценаріїв використання та різних пристроїв.

**5. Видалення фрагментів з Activity. Поняття транзакції фрагментів.**

1. Видалення фрагментів з Activity:

Видалення фрагментів з активності зазвичай виконується за допомогою методу remove або replace об'єкта FragmentManager. Фрагменти додатку можуть бути додані до контейнера в активності, і після цього їх можна видалити.

2. Поняття транзакцій фрагментів:

Транзакції фрагментів представляють собою операції, які виконуються над фрагментами в активності. Транзакції дозволяють додавати, видаляти, замінювати або виконувати інші операції з фрагментами.

Щоб розпочати транзакцію фрагментів, використовують метод beginTransaction об'єкта FragmentManager. Після цього можна додавати, видаляти чи замінювати фрагменти, а також виконувати інші дії.

Приклад видалення фрагменту за допомогою транзакції:

Після виклику commit(), операція видалення фрагменту виконується асинхронно.

Транзакції фрагментів дозволяють ефективно управляти фрагментами в активності та виконувати різні операції з ними, включаючи їх видалення, додавання та заміну.

**6. Основні компонувальники інтерфейсу: LinearLayout, FrameLayout, TableLayout, RelativeLayout, ConstraintLayout**.

Основні компонувальники інтерфейсу в операційній системі Android допомагають розміщати та організовувати елементи на екрані додатку. Кожен з них має свої особливості та використовується в залежності від конкретних потреб дизайну інтерфейсу. Давайте розглянемо основні компонувальники:

LinearLayout:

Лінійний компонувальник (LinearLayout) розміщує елементи в одному ряду (горизонтально) або в одній колонці (вертикально).

Використовується для створення простих списків або кнопок в одному ряду, або для розміщення елементів один під одним.

FrameLayout:

Кадровий компонувальник (FrameLayout) - це найпростіший компонувальник, який дозволяє розмістити декілька елементів в одному контейнері, але він відображає лише один елемент, який перекрива інші.

Зазвичай використовується для створення ефекту накладання (наприклад, для відображення фрагментів поверх іншого змісту).

TableLayout:

Табличний компонувальник (TableLayout) дозволяє розміщати елементи в рядках і стовпцях, схоже до таблиці.

Використовується для створення інтерфейсу, в якому елементи розташовані в формі сітки.

RelativeLayout:

Відносний компонувальник (RelativeLayout) дозволяє розміщати елементи відносно інших елементів або відносно батьківського контейнера.

Використовується для створення складних макетів, де позиція елементів залежить від розміщення інших елементів.

ConstraintLayout:

Контейнер з обмеженнями (ConstraintLayout) дозволяє визначати зв'язки між елементами, такі як відстані, відносний розмір, вирівнювання і т. д.

Використовується для створення складних та адаптивних інтерфейсів, які працюють на різних розмірах екрану.

Кожен із цих компонувальників має свої переваги і використовується в залежності від потреб вашого додатку та дизайну інтерфейсу. Вибір компонувальника залежить від специфічних вимог вашого проекту та задач, які ви хочете вирішити.

**7. Компоненти інтерфейсу Android: Button, TextView, EditText, Switch, CheckBox, RadioButton, RadioGroup, ImageView.**

Компоненти інтерфейсу Android - це різні елементи, які можуть бути використані для створення користувацького інтерфейсу в додатках Android. Ось кілька основних компонентів і їх призначення:

Button (Кнопка):

Button - це елемент інтерфейсу, який користувач може натискати для виклику певної дії. Він використовується для створення кнопок, які виконують певні функції чи операції при натисканні.

TextView (Текстове поле):

TextView - це елемент для відображення текстової інформації на екрані. Він використовується для відображення тексту, міток, заголовків тощо.

EditText (Поле для введення тексту):

EditText - це поле, яке дозволяє користувачу вводити текстову інформацію. Використовується для отримання текстового вводу від користувача.

Switch (Перемикач):

Switch - це компонент для включення або вимкнення якоїсь функції або налаштування. Він має два стани: увімкнено і вимкнено.

CheckBox (Прапорець):

CheckBox - це візуальний елемент, який дозволяє користувачу обирати один чи декілька варіантів з числа доступних варіантів.

RadioButton (Перемикач з одним вибором):

RadioButton - це компонент, який дозволяє користувачу обирати один варіант з числа доступних варіантів. Використовується разом із RadioGroup.

RadioGroup (Група перемикачів):

RadioGroup - це контейнер, який групує RadioButton елементи разом. Використовується для створення груп варіантів вибору, де можна вибрати лише один варіант з групи.

ImageView (Зображення):

ImageView - це елемент для відображення зображень або графічних ресурсів на екрані. Використовується для відображення фотографій, іконок, зображень тощо.

Ці компоненти допомагають розробникам створювати різноманітні та інтерактивні користувацькі інтерфейси в додатках для Android. Залежно від конкретних потреб і дизайну вашого додатку, ви можете використовувати ці компоненти для створення відповідного інтерфейсу для користувачів.

**8. Спискові компоненти інтерфейсу: ListView, RecyclerView, Spinner,AutoCompleteTextView.**

Спискові компоненти інтерфейсу в Android дозволяють відображати і взаємодіяти з списками даних. Ось деякі з них:

ListView (Список):

ListView - це елемент інтерфейсу, який дозволяє відображати список елементів в одній колонці, де кожен елемент списку представлений як окремий рядок.

Використовується для створення простих списків даних, таких як список контактів чи повідомлень.

RecyclerView (Спеціальний список):

RecyclerView - це більш потужний та гнучкий компонент для відображення списків даних. Він дозволяє створювати різні макети списків, включаючи горизонтальні списки, сітки та багато інших.

Використовується для створення складних та динамічних списків, де кожен елемент може містити різні типи даних і виглядати по-різному.

Spinner (Випадаючий список):

Spinner - це елемент інтерфейсу, який дозволяє користувачу вибирати один елемент зі списку, відображаючи лише вибраний елемент, а решта доступна після натискання.

Використовується для створення випадаючих списків, таких як вибір країни чи категорії товарів.

AutoCompleteTextView (Автозаповнення):

AutoCompleteTextView - це текстове поле, яке автоматично відображає пропозиції для введення на основі вмісту списку даних.

Використовується для створення полів для введення, де користувач може швидко знайти та вибрати потрібний елемент із доступного списку.

Ці спискові компоненти допомагають створювати інтерфейси, які включають у себе списки даних, і різняться за функціональністю та стилем відображення. Вибір конкретного компонента залежить від ваших потреб та вимог дизайну вашого додатку.

**9. Принципи реалізації списків. Поняття адаптеру. Реалізація адаптерів на базі ArrayAdapter та BaseAdapter.**

Принципи реалізації списків в Android полягають у створенні і відображенні списку елементів на екрані додатку. При цьому важливо використовувати адаптери для забезпечення взаємодії між даними і відображенням на інтерфейсі користувача.

Адаптер (Adapter) - це посередник між даними та інтерфейсом, який відповідає за подачу даних на відображення та управління елементами списку. В Android існують різні типи адаптерів, включаючи ArrayAdapter та власний адаптер на базі BaseAdapter. Давайте розглянемо їх реалізацію:

1. Реалізація адаптера на базі ArrayAdapter:

ArrayAdapter - це простий адаптер, який призначений для списків, де дані можуть бути збережені в масиві чи списку.

Основні кроки створення списку за допомогою ArrayAdapter:

Створити масив чи список даних, які ви бажаєте відобразити.

Створити екземпляр ArrayAdapter, передавши його контекст, ресурс макету для одного рядка списку і ваші дані.

Прикріпити адаптер до списку, наприклад, до ListView.

Налаштувати обробник подій для списку, якщо необхідно обробляти кліки на елементи списку.

2. Реалізація власного адаптера на базі BaseAdapter:

BaseAdapter - це базовий клас для створення власних адаптерів зі спеціальними потребами.

Основні кроки створення списку за допомогою власного адаптера:

Створити клас, який успадковує BaseAdapter і реалізовує методи, такі як getCount, getItem, getItemId і getView.

Визначити власний макет для рядків списку, які будуть використовуватися для відображення даних.

В методі getView вашого адаптера налаштуйте відображення кожного рядка списку на основі даних зі списку.

Прикріпити свій власний адаптер до списку і налаштувати обробник подій.

Власний адаптер на базі BaseAdapter дає більше контролю над відображенням та управлінням даними, але потребує більше коду. Зазвичай використовують ArrayAdapter, коли масив чи список даних простий, і власний адаптер на базі BaseAdapter, коли потрібна складніша обробка даних та макетів.

**що таке Bundle?**

В Android Bundle - це об'єкт, який використовується для передачі та збереження даних між компонентами додатку, такими як активності (Activities), фрагменти (Fragments) та служби (Services). Основна функція Bundle - це збереження та передача даних в серіалізованій формі, яка може бути легко відновлена на іншому компоненті чи деякому іншому місці в додатку.

Основні риси Bundle включають:

Збереження даних: Bundle може зберігати різні типи даних, такі як рядки, цілі числа, булеві значення, об'єкти та інше.

Серіалізація: Дані в Bundle автоматично серіалізуються та десеріалізуються, тобто перетворюються в байтовий потік для передачі між компонентами та знову конвертуються у вихідні значення.

Передача даних між компонентами: Bundle використовується для передачі додаткової інформації в інший компонент, наприклад, при створенні нової активності чи фрагмента.

Інтенти: Bundle додатково може бути включений в інтент (Intent) для передачі даних між компонентами додатку.

У прикладі вище ми створюємо Bundle, додаємо рядок та ціле число, а потім передаємо цей Bundle до іншої активності за допомогою інтенту. У другій активності можна отримати ці дані та використовувати їх для подальшого виконання.