Мобилки ДЗ 19.09

**1. Введение в технологию мобильных систем**

1. **В чем осо­бен­ность мо­бильной опе­раци­он­ной сис­те­мы?**

Пор­та­тив­ные ус­тройства мо­бильной свя­зи (нап­ри­мер, смар­тфо­ны) со­дер­жат две опе­раци­он­ные сис­те­мы:

1) ос­новная прог­рам­мная плат­форма вза­имо­действия с пользо­вате­лем;

2) низ­ко­уров­не­вая опе­раци­он­ная сис­те­ма ре­ально­го вре­мени, об­слу­жива­ющая ра­ди­ообо­рудо­вание. Та­кие низ­ко­уров­не­вые опе­раци­он­ные сис­те­мы у­яз­ви­мы пе­ред вре­донос­ны­ми ба­зовы­ми стан­ци­ями, спо­соб­ны­ми по­лучить кон­троль над мо­бильным ус­тройством.

1. **На­зови­те из­вес­тные сов­ре­мен­ные мо­бильные опе­раци­он­ные сис­те­мы.**

IOS и Android ( на ядре линукс, владелец гугл)

1. **Ка­кие уров­ни су­щес­тву­ют в опе­раци­он­ной сис­те­ме Android?**

1) уро­вень яд­ра – Драйверы оборудования, управление питанием, управление процессами, управление памятью;

2) уро­вень биб­ли­отек и сре­ды вы­пол­не­ния – Системная библиотека bionic, мультимедиа WebKit, графика (Free Type, OpenGL, SGL), SQLite; Рабочая среда андроида - Библиотеки Android, Виртуальная машина ( DVM, dalvik, virtual machine)

3) уро­вень фреймвор­ка при­ложе­ний (application framework) – диспетчер активности, пакетов, окон, ресурсов, телефонии, местоположения, уведомлений, контент-провайдеры, система представлений

4)уро­вень при­ложе­ний - встроенные приложения (контакты, браузер), сторонние приложения, приложения разраба.

1. **Оха­рак­те­ризуйте фун­кции каж­до­го из уров­ней опе­раци­он­ной сис­те­мы Android.**
   1. **уровень ядра** - Это основной уровень операционной системы, который управляет аппаратными ресурсами устройства. Он отвечает за управление памятью, процессами, файловой системой и сетевыми соединениями.
   2. **уровень библиотек и среды выполнения** содержит набор библиотек, которые обеспечивают функциональность для приложений. Он включает в себя как стандартные библиотеки, так и сторонние.

- WebKit для работы с веб-контентом.

- OpenGL для графики.

- SQLite для работы с базами данных.

* 1. **уровень фреймворка приложений –** Этот уровень предоставляет API и инструменты для разработки приложений. Он управляет жизненным циклом приложений, ресурсами, пользовательским интерфейсом и взаимодействием между компонентами.
  2. **уровень приложений** Этот уровень включает в себя пользовательские приложения, такие как мессенджеры, браузеры, игры и т.д. Приложения взаимодействуют с пользователем и используют API Android для доступа к системным ресурсам.

1. **Ка­ковы фун­кции DVM?**

(Dalvik Virtual Machine)

- Исполнение байт-кода: DVM выполняет байт-код, скомпилированный из Java-программ. Это позволяет приложениям работать на устройствах с Android.

- Управление памятью: DVM использует сборщик мусора для управления памятью, что помогает избежать утечек памяти и оптимизировать использование ресурсов.

- Оптимизация: DVM обеспечивает оптимизацию выполнения приложений, включая компиляцию в реальном времени (JIT) для повышения производительности.

- Поддержка многопоточности: DVM поддерживает многопоточность, что позволяет приложениям выполнять несколько задач одновременно.

1. **Что на­зыва­ет­ся SDK?**

***Android SDK(Software Development Kit)*** — инс­тру­мен­тальные средс­тва раз­ра­бот­ки мо­бильных при­ложе­ний для опе­раци­он­ной сис­те­мы Android.

1. **Что вхо­дит в сос­тав Android SDK?**

Android SDK (Software Development Kit) включает в себя:

- API библиотеки: Наборы библиотек для разработки приложений.

- Инструменты для разработки: Android Studio, эмуляторы, инструменты командной строки.

- Документация: Полные руководства и примеры по использованию API и инструментов.

- Примеры кода: Примеры приложений и кода для обучения разработчиков.

- Эмуляторы: Средства для тестирования приложений на различных устройствах и конфигурациях.

1. Android Studio - интегрированная среда разработки (IDE) для создания приложений под Android.
2. Android Debug Bridge (adb) - инструмент для взаимодействия с устройством Android, таким как установка и удаление приложений, отладка и многое другое.
3. Android Emulator - эмулятор Android-устройств, позволяющий запускать и тестировать приложения без реального устройства.
4. Android Platform Tools - набор инструментов, таких как adb, fastboot и другие, для работы с Android-устройствами.
5. Android SDK Tools - основные инструменты разработки для создания и сборки приложений под Android.
6. Библиотеки и API - наборы библиотек и API для разработки приложений под Android, такие как Android Support Library, Google Play Services и другие.
7. Документация и примеры кода - руководства по разработке, API-документация, примеры кода и другие ресурсы для помощи разработчикам создавать приложения под Android.
8. **Где про­ис­хо­дит от­ладка и тес­ти­рова­ние при­ложе­ния для And­roid?**

В эмуляторе Android Studio, потом на реальных устройствах

1. **По­чему нуж­но ис­пользо­вать эму­лятор Android-ус­тройства?**

Эму­лятор поз­во­ля­ет от­ла­живать и тес­ти­ровать при­ложе­ния в ре­альной сре­де вы­пол­не­ния без не­об­хо­димос­ти их ус­та­нов­ки на фи­зичес­кое ус­тройство. Так как но­вое при­ложе­ние мо­жет ра­ботать не­ста­бильно, то сна­чала ус­тройство нуж­но за­пус­тить в эму­лято­ре, и только за­тем на ре­альном ус­тройстве. Как в ре­альном, так и в вир­ту­альном ус­тройстве су­щес­тву­ет опас­ность раз­ру­шить файло­вую сис­те­му, нап­ри­мер, при неп­ра­вильном прек­ра­щении ра­боты (сбое пи­тания и др.).

1. **Для че­го каж­до­му при­ложе­нию Android прис­ва­ива­ет­ся уни­кальный иден­ти­фика­тор UID?**

UID (Unique Identifier) обеспечивает:

- Безопасность: Каждое приложение работает в своем собственном контексте, что предотвращает несанкционированный доступ к данным других приложений.

- Изоляция: UID изолирует приложения друг от друга, что уменьшает риск конфликтов и ошибок.

- Управление ресурсами: Система может отслеживать использование ресурсов каждым приложением и управлять ими более эффективно.

1. **Как соз­да­ет­ся бе­зопас­ная сре­да для при­ложе­ния Android?**

Безопасная среда для приложения Android создается благодаря:

- Изоляции приложений: Каждое приложение работает в своем собственном процессе и имеет уникальный UID.

- Разрешениям: Приложения должны запрашивать разрешения на доступ к определенным ресурсам, таким как камера или местоположение.

- Песочнице (Sandbox): Приложения работают в песочнице, что ограничивает их доступ к системным ресурсам и данным других приложений.

- Шифрованию данных: Android поддерживает шифрование данных для защиты конфиденциальной информации.

1. **Для че­го ис­пользу­ет­ся JRE?**

JRE (Java Runtime Environment) используется для:

- Выполнения Java-программ: JRE предоставляет среду выполнения для запуска Java-приложений, включая библиотеки и виртуальную машину (JVM).

- Поддержки кросс-платформенности: JRE позволяет запускать Java-программы на разных платформах без необходимости компиляции под каждую из них.

- Обеспечения стандартного API: JRE включает стандартные библиотеки Java, которые разработчики могут использовать при создании приложений.

**2.Структура и принципы функционирования мобильного приложения**

1. **Что на­зыва­ет­ся *Ак­тивностью* при­ложе­ния?**

ви­димая часть при­ложе­ния (эк­ран, ок­но, фор­ма), ко­торая от­ве­ча­ет за отоб­ра­жение гра­фичес­ко­го ин­терфейса пользо­вате­ля.

1. **Ка­кие ком­по­нен­ты при­ложе­ния на­зыва­ют сер­ви­сами?**

сер­висные ком­по­нен­ты, ко­торые ра­бота­ют в фо­новом ре­жиме, за­пус­кая уве­дом­ле­ния, об­новляя ис­точни­ки дан­ных и ви­димые ак­тивнос­ти. *Сер­ви­сы* ис­пользу­ют­ся для ре­гуляр­ных опе­раций, ко­торые дол­жны про­дол­жаться да­же тог­да, ког­да ак­тивнос­ти при­ложе­ния не на пе­ред­нем пла­не. *Сер­ви­сы* (служ­бы) не име­ют гра­фичес­ко­го ин­терфейса пользо­вате­ля.

1. **Для че­го слу­жат *Кон­тент-про­вайде­ры*?**

слу­жат для пре­дос­тавле­ния дос­ту­па к дан­ным прог­раммы дру­гим при­ложе­ни­ям.

1. **Ка­кие ком­по­нен­ты на­зыва­ют­ся *На­мере­ни­ями* и для че­го они слу­жат?**

сис­те­ма пе­реда­чи со­об­ще­ний меж­ду при­ложе­ни­ями.

1. **Для че­го слу­жит *При­ем­ник ши­роко­веща­тельных со­об­ще­ний*?**

для по­луче­ния ин­форма­ции о внеш­них со­быти­ях и ре­ак­ции на них. Большинс­тво та­ких со­об­ще­ний по­рож­да­ют­ся сис­те­мой, нап­ри­мер, со­об­ще­ние о том, что эк­ран от­клю­чил­ся или низ­кий за­ряд ба­тареи.

1. **Что на­зыва­ет­ся *Вид­же­том*?**

ви­зу­альные прог­рам­мные ком­по­нен­ты, ко­торые мож­но до­бав­лять на до­маш­ний эк­ран. Этот осо­бый вид ши­роко­веща­тельных при­ем­ни­ков, поз­во­ля­ющий соз­да­вать ди­нами­чес­кие, ин­те­рак­тивные ком­по­нен­ты, ко­торые пользо­вате­ли мо­гут встра­ивать в свои до­маш­ние эк­ра­ны.

1. **Для че­го ис­пользу­ет­ся сис­те­ма пользо­вательских уве­дом­ле­ний?**

поз­во­ля­ют сиг­на­лизи­ровать о чем-ли­бо, не об­ра­щая на се­бя вни­мание или не пре­рывая ра­боту те­кущей *Ак­тивнос­ти*.

1. **Как оце­нива­ет­ся опе­раци­он­ной сис­те­мой важ­ность каж­до­го за­пущен­но­го про­цес­са?**

Важ­ность каж­до­го за­пущен­но­го про­цес­са оце­нива­ет­ся ОС преж­де все­го с точ­ки зре­ния пользо­вате­ля. Так, нап­ри­мер, ОС мо­жет за­вер­шить про­цесс с не­види­мыми *Ак­тивнос­тя­ми*, ко­торые бо­лее не отоб­ра­жа­ют­ся на эк­ра­не, од­на­ко ни­ког­да не унич­то­жит ви­димый пользо­вате­лем ком­по­нент про­цес­са.

1. **Ка­кой про­цесс на­зыва­ют ак­тивным?**

***Ак­тивный про­цесс*** (*Foreground Process*) — про­цесс, с ко­торым в дан­ный мо­мент ра­бота­ет пользо­ватель. Так­же про­цесс счи­та­ет­ся ак­тивным, ес­ли он име­ет объект Service и пользо­вате­лем вы­пол­ня­ет­ся один из ме­тодов об­ратно­го вы­зова, ко­торый оп­ре­делен в этом объек­те.

1. **Ка­кой про­цесс на­зыва­ют фо­новым?**

***Фо­новый про­цесс*** (*Background Process*) — про­цесс, в ко­тором вы­пол­ня­ет­ся де­ятельность, ко­торая в дан­ный мо­мент не вид­на пользо­вате­лю.

1. **Ка­кой про­цесс на­зыва­ют хо­лос­тым?**

***Пус­той про­цесс*** (*Empty Process*) — про­цесс, ко­торый не со­дер­жит ак­тивных ком­по­нен­тов при­ложе­ния

1. **Ка­кой про­цесс на­зыва­ют ви­димым?**

***Ви­димый*** ***про­цесс*** (*Visible Process*) — про­цесс, не име­ющий ни­каких при­ори­тет­ных ком­по­нен­тов.

1. **Ка­кой про­цесс на­зыва­ют сер­висным?**

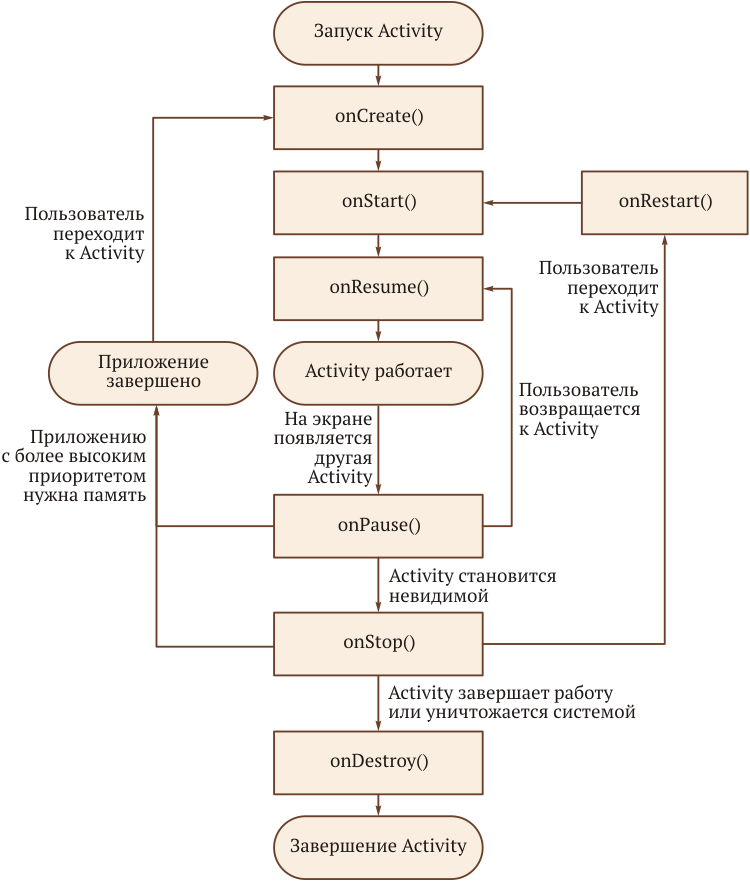
***Сер­висный про­цесс*** (*Service Process*) — про­цесс, в ко­тором вы­пол­ня­ет­ся сер­вис (служ­ба) и ко­торый не от­но­сит­ся к ак­тивным и ви­димым про­цес­сам.

1. **Для че­го ис­пользу­ет­ся *Ма­нифест* при­ложе­ния?**

Ма­нифест поз­во­ля­ет опи­сывать струк­ту­ру и ме­тадан­ные при­ложе­ния, его ком­по­нен­ты и тре­бова­ния. В файле ма­нифес­та ука­зыва­ет­ся имя Java-па­кета при­ложе­ния (имя па­кета ис­пользу­ет­ся как уни­кальный иден­ти­фика­тор при­ложе­ния), раз­ре­шения, ко­торы­ми дол­жно об­ла­дать при­ложе­ние для об­ра­щения к за­щищен­ным час­тям API (нуж­но для вза­имо­действия с дру­гими при­ложе­ни­ями), биб­ли­оте­ки, не­об­хо­димые для вы­пол­не­ния это­го при­ложе­ния, и др.

1. **Что за­писы­ва­ет­ся в файл *Ма­нифес­та*?**

В ма­нифес­те хра­нит­ся опи­сание ком­по­нен­тов при­ложе­ния, а имен­но: *Ак­тивнос­тей*, *Сер­ви­сов*, *Ис­точни­ков дан­ных* (*Content Provider*) и др.

1. **В ка­ких воз­можных сос­то­яни­ях мо­жет на­ходиться *Ак­тивность*? В чем осо­бен­ность каж­до­го сос­то­яния?**
   1. *Ак­тивное* (*active* или *running*). *Ак­тивность* на­ходит­ся на пе­ред­нем пла­не (на вер­ши­не сте­ка) и име­ет воз­можность вза­имо­действо­вать с пользо­вате­лем.
   2. *При­ос­та­нов­ленное* (*paused*)*. Ак­тивность* мо­жет быть вид­на на эк­ра­не, но не мо­жет вза­имо­действо­вать с пользо­вате­лем.
   3. *Ос­та­нов­ленное* (*stopped*). *Ак­тивность* не­види­ма, но она на­ходит­ся в па­мяти, сох­ра­няя ин­форма­цию о сво­ем сос­то­янии.
   4. *Не­ак­тивное* (*inactive*). Ра­бота *Ак­тивнос­ти* за­вер­ше­на, и она на­ходит­ся в не­ак­тивном сос­то­янии. Та­кие *Ак­тивнос­ти* уда­ля­ют­ся из сте­ка
2. **Пе­речис­ли­те ме­тоды, вы­зыва­емые сис­те­мой для уп­равле­ния жиз­ненным цик­лом *Ак­тивнос­ти*.**
3. **Опи­шите ме­тоды клас­са Application.**
   1. Ме­тод onCreate() — вы­зыва­ет­ся при соз­да­нии при­ложе­ния, пе­ре­оп­ре­деля­ет­ся для соз­да­ния и ини­ци­али­зации свойств, в ко­торых хра­нят­ся сос­то­яния при­ложе­ния или гло­бальные ре­сур­сы.
   2. Ме­тод onTerminate() — мо­жет быть выз­ван при преж­девре­мен­ном за­вер­ше­нии ра­боты при­ложе­ния. Од­на­ко, ес­ли при­ложе­ние зак­ры­ва­ет­ся яд­ром, что­бы ос­во­бодить ре­сур­сы для дру­гих прог­рамм, onTerminate выз­ван не бу­дет.
   3. Ме­тод onLowMemory() — пре­дос­тавля­ет воз­можность при­ложе­ни­ям ос­во­бодить до­пол­ни­тельную па­мять, ког­да ОС не хва­та­ет ре­сур­сов. Этот ме­тод пе­ре­оп­ре­деля­ет­ся для то­го, что­бы очис­тить кэш или ос­во­бодить не­нуж­ные в дан­ный мо­мент ре­сур­сы.
   4. Ме­тод onConfigurationChanged() — пе­ре­оп­ре­деля­ет­ся, ес­ли не­об­хо­димо от­сле­живать из­ме­нения кон­фи­гура­ции на уров­не при­ложе­ния, нап­ри­мер, та­кие как по­ворот эк­ра­на, зак­ры­тие выд­вижной кла­ви­ату­ры ус­тройства и др.
4. **Для че­го ис­пользу­ет­ся класс android.context.Context?**

Класс android.context.Context пред­став­ля­ет со­бой ин­терфейс для дос­ту­па к гло­бальной ин­форма­ции об ок­ру­жении при­ложе­ния — абс­трак­тный класс, ре­али­зация ко­торо­го обес­пе­чива­ет­ся сис­те­мой Android. Класс Context поз­во­ля­ет по­лучить дос­туп к спе­цифич­ным для дан­но­го при­ложе­ния ре­сур­сам и клас­сам, а так­же для вы­зова опе­раций на уров­не при­ложе­ния, та­ких как за­пуск *Ак­тивнос­тей*, от­прав­ка ши­роко­веща­тельных со­об­ще­ний, по­луче­ние *На­мере­ний* и др. Дан­ный класс так­же яв­ля­ет­ся ба­зовым для клас­сов Activity, Application и Service.