Android Java Specifics / Basic IO

Freie Universität Berlin

- Dalvik virtual machine
  - Registermaschine anstelle einer Stackmaschine
  - Optimiert für geringen Speicherverbrauch
  - Umwandlung von .class Dateien in .dex-Format mit Hilfe von *dx*
  - Jedes Programm läuft als eigener Prozess mit eigener Dalvik VM
  - Sprachumfang mindestens Java 1.5
- Programmierempfehlungen für hohe Geschwindigkeit
  - Möglichst wenig Objekte erzeugen
  - Enums, getter und setter vermeiden
  - Kein float verwenden
  - Siehe: http://d.android.com/guide/practices/design/performance.html

Android Java Specifics / Basic IO

Freie Universität Berlin

- Vier Möglichkeiten um Anwendungsdaten zu speichern
  - *Preferences*
    - Lesen und Schreiben von Schlüssel-Wert-Paaren
    - Kein Zugriff für fremde Programme
  - Dateien
    - Intern im Gerät oder auf externem Speichermedium
    - Standardmäßig kein Zugriff für fremde Programme
  - Datenbanken
    - SQLite – Untermenge von SQL
    - Zugriff auf Datenbank nur für erzeugendes Programm
  - Network
    - Siehe Networking-Gruppe

Android Java Specifics / Basic IO

Freie Universität Berlin

- Das *SharedPreferences*-Interface
  - public abstract SharedPreferences **getSharedPreferences**(String name, int mode)
    - *name*: beliebiger Text zur Identifikation
    - *mode*: MODE_PRIVATE, MODE_WORLD_READABLE oder MODE_WORLD_WRITEABLE
    - Preferences-Datei wird automatisch erzeugt falls nicht vorhanden
  - Zugriff auf Werte über verschiedene Methoden
    - abstract boolean **getBoolean**(String key, boolean defValue)
    - abstract int **getInt**(String key, int defValue)
    - ...
  - Für Änderungen Objekt vom Typ SharedPreferences.Editor notwendig
    - abstract SharedPreferences.Editor **edit**()
    - abstract SharedPreferences.Editor **putBoolean**(String key, boolean value)
    - abstract SharedPreferences.Editor **putInt**(String key, int value)
    - ...
  - Am Ende Aufruf zum Speichern nötig
    - abstract boolean **commit**()

Einfaches Beispiel ohne Exception-Behandlung:

```
...
// Preferences auslesen
SharedPreferences settings = Context.getSharedPreferences("MyPreferences",
    MODE_PRIVATE);
boolean silent = settings.getBoolean("silentMode", false);
...
// Preferences speichern
SharedPreferences.Editor editor = settings.edit();
editor.putBoolean("MyPreferences", true);
editor.commit();
...
```

Android Java Specifics / Basic IO

Freie Universität Berlin

- Zugriff auf Dateien
  - Context.openFileInput() zum Öffnen einer Datei
    - public abstract FileInputStream **openFileInput** (String name)
    - name: Name und Pfad der gewünschten Datei
  - Context.openFileOutput() zum Schreiben in eine Datei
    - public abstract FileOutputStream **openFileOutput** (String name, int mode)
    - name: Name und Pfad der gewünschten Datei
    - mode: MODE_PRIVATE, MODE_APPEND, MODE_WORLD_READABLE oder MODE_WORLD_WRITEABLE
  - Kein Zugriff auf Dateien anderer Programme

Android Java Specifics / Basic IO

Freie Universität Berlin

Einfaches Beispiel ohne Exception-Behandlung:

```
...
// Datei auslesen
FileInputStream fis = Context.openFileInput("MyFile");
while (fis.available() > 0) {
      System.out.println(fis.read());
}
fis.close();
...
// Datei abspeichern
FileOutputStream fos = Context.openFileOutput("MyFile", MODE_PRIVATE);
fos.write("foobar".getBytes());
fos.close();
...
```

Android Java Specifics / Basic IO

Freie Universität Berlin

- Zugriff auf Datenbanken mit SQLiteDatabase-Objekt
  - Temporäre Datenbank erzeugen
    - public static SQLiteDatabase **create**(SQLiteDatabase.CursorFactory factory)
    - Datenbank wird im Speicher gehalten und beim Schließen gelöscht
  - Persistente Datenbank öffnen
    - public static SQLiteDatabase **openDatabase**(String path, SQLiteDatabase.CursorFactory factory, int flags)
  - SQL-Befehle (CREATE, DELETE, INSERT)
    - public void **execSQL**(String sql)
    - Nur ein Statement pro Aufruf möglich
  - SQL-Abfragen (zwei Möglichkeiten)
    - public Cursor **query**(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)
    - public Cursor **rawQuery**(String sql, String[] selectionArgs)
  - Transaktionen und viele weitere Funktionen vorhanden

Einfaches Beispiel ohne Exception-Behandlung:

```
…
// Wert in Datenbank einfügen
SQLiteDatabase db = openDatabase("MyDB", null, OPEN_READWRITE);
db.execSQL("INSERT INTO myTable (foo) VALUES ('bar');");
…
// Wert aus Datenbank lesen
Cursor cur = rawQuery("SELECT foo FROM myTable WHERE foo='bar';", null);
int columnIndex = cur.getColumnIndexOrThrow("foo");
cur.moveToFirst();
for (int i = 0; i < cs.getCount(); ++i) {
      System.out.println(cur.getString(columnIndex));
      cur.moveToNext();
}
cur.close();
db.close();
…
```