2. Standortbezogene Dienste

Lernziele:

- Location API von Android kennen und einsetzen können,
- Umgang mit Kartendarstellungen in Android beherrschen,
- allgemeine Konzepte der Lokalisierung, insbesondere GPS, kennen und
- die wichtigsten Elemente der Geoinformatik für standortbezogene Dienste anwenden können.

Location Based Services

- Ermittlung der aktuellen Position eines Nutzers
- Ausnutzung der Positionsdaten für eine Dienstleistung

Lokalisierung mit Hilfe von:

- Satelliten-System
- Funknetze
- weiteren Sensoren

Beispielangebote ohne reine Navigation

- ADAC Mobile Service: Pannenhilfe, Parkleitsystem, Staumeldungen, Reiseinfo
- Notfon D: Handy-Ortung bei Autopannen
- Vitaphone: Telemedizin-System, Notruf (GPS)
- Track your kid: Aufenthaltsort der eigenen Kinder ermitteln (GSM)
- Track your truck: Aufenthaltsort von Fahrzeugen ermitteln (GPS oder GSM)

- Mobiloco: Freunde lokalisieren, Einkaufsmöglichkeiten in der Nähe finden (GSM)

- Virtic: Arbeitszeiterfassung z.B. auf Großbaustellen, Ortung der Mitarbeiter

Location Manager

- Klasse `android.location.LocationManager`
- Einstiegspunkt für Dienste zur Lokalisierung
- Ermittlung der verfügbaren Dienste zur Lokalisierung
- Standortermittlung
- Registrierung von Listenern

Zugriff auf den Location Manager

```
import android.location.LocationManager;

LocationManager lm = (LocationManager)
    getSystemService(Context.LOCATION_SERVICE);
```

Registrierung von Listenern

Beim `LocationManager` für:

- Das Erkennen von Positionsänderungen,
- die Annäherung an Punkte und
- das Verlassen von Bereichen.

Ermittlung eines `LocationProvider`

Methoden der Klasse `android.location.LocationManager`:

- `public List<String> getAllProviders ()`

  Liste mit Bezeichnungen für alle prinzipiell zur Verfügung stehenden Lokalisierungsdienste, inklusive Dienste ohne Berechtigung und Dienste, die z. Z. nicht zur Verfügung stehen.

- `public LocationProvider getProvider (String name)`

  Referenz auf den Lokalisierungsdienst mit der Bezeichnung `name`.

- `public List<String> getProviders (boolean enabledOnly)`

  Liste mit Bezeichnungen für Lokalisierungsdienste, nur Dienste mit Berechtigung und nur verfügbare (falls `enabledOnly` gleich `true`)

- `public List<String> getProviders (Criteria criteria, boolean enabledOnly)`

  Lokalisieurngsdienste, die vorgegebene Kriterien erfüllen und verfügbar sind (falls `enabledOnly` gleich `true`)

- `public String getBestProvider (Criteria criteria, boolean enabledOnly)`

  Lokalisierungsdienst, der am besten zu den vorgegebenen Kriterien passt und verfügbar ist (falls `enabledOnly` gleich `true`)

Vordefinierte Location Provider

String-Konstanten in android.location.LocationManager:

- `GPS_PROVIDER`

  String "gps"

  Benötigt Uses-Permission: `android.permission.ACCESS_FINE_LOCATION`

- `NETWORK_PROVIDER`

  String "network"

  Benötigt Uses-Permission: `android.permission.ACCESS_FINE_LOCATION` oder `android.permission.ACCESS_COARSE_LOCATION`

Kriterien für die Provider-Auswahl

Klasse `android.location.Criteria`

Eigenschaften für Auswahl:

- Genauigkeit:
  `public void setAccuracy (int accuracy)`

  Konstanten für die Auswahl in `Criteria`: `ACCURACY_FINE`, `ACCURACY_COARSE`

- Stromverbrauch:
  `public void setPowerRequirement (int level)`

  maximaler Stromverbrauch, Konstanten: `NO_REQUIREMENT`, `POWER_LOW`, `POWER_MEDIUM`, `POWER_HIGH`

- Höhe:
  `public void setAltitudeRequired (boolean altitudeRequired)`

- Bewegungsrichtung:
  `public void setBearingRequired (boolean bearingRequired)`

- Geschwindigkeit:
  `public void setCostAllowed (boolean costAllowed)`

- Kosten:
  `public void setSpeedRequired (boolean speedRequired)`

Location Provider

Abstrakte Klasse `android.location.LocationProvider`:

- repräsentiert allgemein einen Dienst bzw. eine Möglichkeit zur Standortbestimmung
- spezialisierte Unterklassen für die spezifischen Dienste
- periodische Standortermittlung
- jeder Provider hat bestimmte Kriterien, die die Benutzbarkeit definieren
- abstrakte Methoden zur Bestimmung der Ausprägung der Kriterien

Konstanten zur Beschreibung des Zustands eines `LocationProvider`:

- `AVAILABLE`: Provider ist verfügbar und kann Daten liefern
- `TEMPORARILY_UNAVAILABLE`: zur Zeit nicht verfügbar, wird aber voraussichtlich bald wieder zur Verfügung stehen
- `OUT_OF_SERVICE`: nicht verfügbar

Zustand wird über `LocationListener` übermittelt!

Positionsbestimmung

Klasse `android.location.LocationManager`:

- `public Location getLastKnownLocation (String provider)`

  direkte Abfrage der letzten Position für `provider`

- `public void requestLocationUpdates (String provider, long minTime, float minDistance, LocationListener listener)`

  Registrierung eines Listeners, um regelmäßig Positionsdaten zu erhalten

- `public void removeUpdates (LocationListener listener)`

  Abmelden des Listeners

Location Listener

Schnittstelle `android.location.LocationListener`:

- `public abstract void onLocationChanged (Location location)`

  Übermittlung der neuen Position in `location`

- `public abstract void onProviderDisabled (String provider)`

  Benutzer hat Lokalisierungsdienst `provider` abgestellt

- `public abstract void onProviderEnabled (String provider)`

  Benutzer hat Lokalisierungsdienst `provider` angestellt

- `public abstract void onStatusChanged (String provider, int status, Bundle extras)`

  Statusänderung des Lokalisierungsdienstes `provider`

  neuer Zustand ist `status` (in `LocationProvider` definierte Konstanten)

  zusätzliche dienstspezifische Informationen in `extras`, z. Z. nur "satellites" für die Anzahl der Satelliten, auf der die Positiosbestimmung beruht, weitere Statusinformationen durch andere Schnittstellen

Location

Klasse `android.location.Location`, beschreibt eine einzelne Position mit folgenden Eigenschaften:

- Breitengrad: `public double getLatitude ()`
- Längengrad: `public double getLongitude ()`
- Höhe: `public double getAltitude ()`
- Bewegungsrichtung: `public float getBearing ()`
- Geschwindigkeit: `public float getSpeed ()`

- Uhrzeit: `public long getTime ()`
- Genauigkeit: `public float getAccuracy ()`
- Dienstspezifische Zusatzinformationen:
  `public Bundle getExtras ()`

Formatierung von Koordinaten

Mit `android.location.Location`:

Konstanten für die Formatierung:

- `FORMAT_DEGREES`: Constant used to specify formatting of a latitude or longitude in the form "[+-]DDD.DDDDD" where D indicates degrees.
- `FORMAT_MINUTES`: Constant used to specify formatting of a latitude or longitude in the form "[+-]DDD:MM.MMMMM" where D indicates degrees and M indicates minutes of arc (1 minute = 1/60th of a degree).

- `FORMAT_SECONDS`: Constant used to specify formatting of a latitude or longitude in the form "DDD:MM:SS.SSSSS" where D indicates degrees, M indicates minutes of arc, and S indicates seconds of arc (1 minute = 1/60th of a degree, 1 second = 1/3600th of a degree).

Klassenmethoden für die Umrechnung:

- von `double` in `String`:
  ```
  public static String convert (double coordinate, int outputType)
  ```

- von `String` in `double`:
  ```
  public static double convert (String coordinate)
  ```

Entfernungsmessung und Zielrichtung

- Entfernung zu einer Zielposition (Instanzmethode):
  ```
  public float distanceTo (Location dest)
  ```
  `Location` bietet entsprechende Setter-Methoden an (siehe API)

- Entfernung zwischen zwei Positionen (Klassenmethode):
  ```
  public static void distanceBetween (
  double startLatitude, double startLongitude,
  double endLatitude, double endLongitude,
  float[] results)
  ```
  `result[0]` enthält die Entfernung in Metern.
  `result[1]` und `result[2]` sind optional, geben Bewegungsrichtung (bearing) am Start bzw. Zielpunkt an.

- Richtung zu einer Position (Instanzmethode):
  `public float bearingTo (Location dest)`

Genauere Statusinformationen bei GPS

1. Registrierung eines `GpsStatus.Listener` über den `LocationManager`:
   `public boolean addGpsStatusListener (GpsStatus.Listener listener)`

2. Empfang von Änderungsereignissen über die Schnittstelle `android.location.GpsStatus.Listener`:
   `public abstract void onGpsStatusChanged (int event)`

   Vordefinierte Konstanten für `event` in `android.location.GpsStatus`:

   - `GPS_EVENT_FIRST_FIX`: Event sent when the GPS system has received its first fix since starting.
   - `GPS_EVENT_SATELLITE_STATUS`: Event sent periodically to report GPS satellite status.

- `GPS_EVENT_STARTED`: Event sent when the GPS system has started.
- `GPS_EVENT_STOPPED`: Event sent when the GPS system has stopped.

3. Innerhalb von `onGpsStatusChanged()` die folgende Instanzmethode des `LocationManager` aufrufen:
   `public GpsStatus getGpsStatus (GpsStatus status)`

4. android.location.GpsStatus stellt dann zur Verfügung:

   - Maximale Anzahl an Satelliten:
     `public int getMaxSatellites ()`
   - Liste mit aktuellen Satelliten:
     `public Iterable<GpsSatellite> getSatellites ()`
   - Zeit bis erste Position verfügbar nach Wiederanlauf von GPS:
     `public int getTimeToFirstFix ()`

Zustand eines Satelliten

Klasse `android.location.GpsSatellite`:

- `public float getAzimuth ()`

  Horizontalwinkel (Himmelsrichtung), in der sich der Satellit relativ zum Standort befindet

- `public float getElevation ()`

  Vertikalwinkel der Satellitenposition relativ zum Standort

- `public int getPrn ()`

  Returns the PRN (pseudo-random number) for the satellite.

- `public float getSnr ()`

  Returns the signal to noise ratio for the satellite.

- `public boolean hasAlmanac ()`

  Liefert `true`, wenn die Gps-Engine über die Bahndaten (Almanach) des Satelliten verfügt.

- `public boolean hasEphemeris ()`

  Liefert `true`, wenn die Gps-Engine über die Bahndaten (Ephemeride) des Satelliten verfügt.

- `public boolean usedInFix ()`

  Liefert `true`, wenn der Satellit in die letzte Positionsberechnung einging.

GPS-Daten nach dem NMEA-Standard

- National Marine Electronics Association (NMEA)
- US-Vereinigung von Elektronikherstellern und -händlern der Schiffsindustrie
- NMEA definiert Standards für die Übertragung von Daten im maritimen Bereich
- wichtigste Anwendung: Positionsdaten
- Standard: NMEA 0183

Empfang von NMEA 0183-Datensätzen

1. Registrierung eines `GpsStatus.NmeaListener` über den `LocationManager`:
   `public boolean addNmeaListener (GpsStatus.NmeaListener listener)`

2. Empfang von NMEA-Datensätzen über die Schnittstelle
   `android.location.GpsStatus.NmeaListener`:
   `public abstract void onNmeaReceived (long timestamp, String nmea)`

Exkurs: `PendingIntent`

- Klasse `android.app.PendingIntent`
- kapselt einen Intent und eine Aktion um:
  - eine Activity zu starten,
  - einen Broadcast-Intent auszulösen oder
  - einen Service zu starten.
- Beispiel: Erzeugung eines Pending-Intents für Broadcast:
  ```
  public static PendingIntent getBroadcast (Context context, int
  requestCode, Intent intent, int flags)
  ```

Bereichskontrolle

1. Mit `LocationListener` einen `PendingIntent` registrieren:
   ```
   public void addProximityAlert (double latitude, double longitude,
   float radius, long expiration, PendingIntent intent)
   ```

2. Kontrolle erfolgt über eine Zeitdauer von `expiration` Millisekunden oder bei -1 unbegrenzt

3. Kontrollbereich ist die Kreisfläche mit `radius` um die durch `latitude` und `longitude` angegebene Position

4. Der ausgelöste Intent hat ein `boolean`-Extra mit dem Schlüssel `KEY_PROXIMITY_ENTERING`

Der zugehörige Wert ist `true`, wenn der Kontrollbereich betreten wird.

5. `intent` definiert die Aktion, die ausgelöst wird und den damit verbundenen Intent, wenn der Kontrollbereich betreten oder verlassen wird.

6. **Beispiel**: siehe Homepage, Auszug aus H. Mosemann, M. Kose, *Android*, Hanser, 2009.

Georeferenzierung

- *Georeferenzierung* ist die Zuweisung von raumbezogenen Informationen zu einem Datensatz.
- *Adresskodierung*: Zuweisung einer Postanschrift zu Geokoordinaten, auch als *Reverse Geocoding* bezeichnet.
- *Geokodierung*: Zuweisung von Koordinaten zu einer Postanschrift (*Forward Geocoding*)
- notwendig: `Locale`-Instanz, siehe Klasse `java.util.Locale`
- Beispiel: `Locale.GERMANY`

Geocoder

Klasse `android.location.Geocoder`:

- Konstruktor:
  `public Geocoder (Context context, Locale locale)`

- Beispiel:

```
Geocoder geocoder = new Geocoder(getApplicationContext(),
                                 Locale.getDefault());
```

- Methoden für Forward Geocoding und Reverse Geocoding

Reverse Geocoding

Klasse `android.location.Geocoder`:

- `public List<Address> getFromLocation (double latitude, double longitude, int maxResults)`

  Liefert `Address`-Instanzen, zu der durch `latitude` und `longitude` beschriebenen Position.

  Zugriff auf einen Netzwerkdienst, evtl. hohe Latenzzeiten

- Repräsentation der Adressen durch Instanzen der Klasse `android.location.Address`

Forward Geocoding

Klasse `android.location.Geocoder`:

- `public List<Address> getFromLocationName (String locationName, int maxResults)`

  Liefert `Address`-Instanzen zu der durch `locationName` beschriebenen Adresse.

  Die `Address`-Instanzen beinhalten Getter-Methoden für Koordinaten!

- ```
  public List<Address> getFromLocationName (
  String locationName, int maxResults,
  double lowerLeftLatitude, double lowerLeftLongitude,
  double upperRightLatitude, double upperRightLongitude)
  ```

  Spezifischere Suche durch Beschränkung auf ein umgebendes Rechteck.
  ```