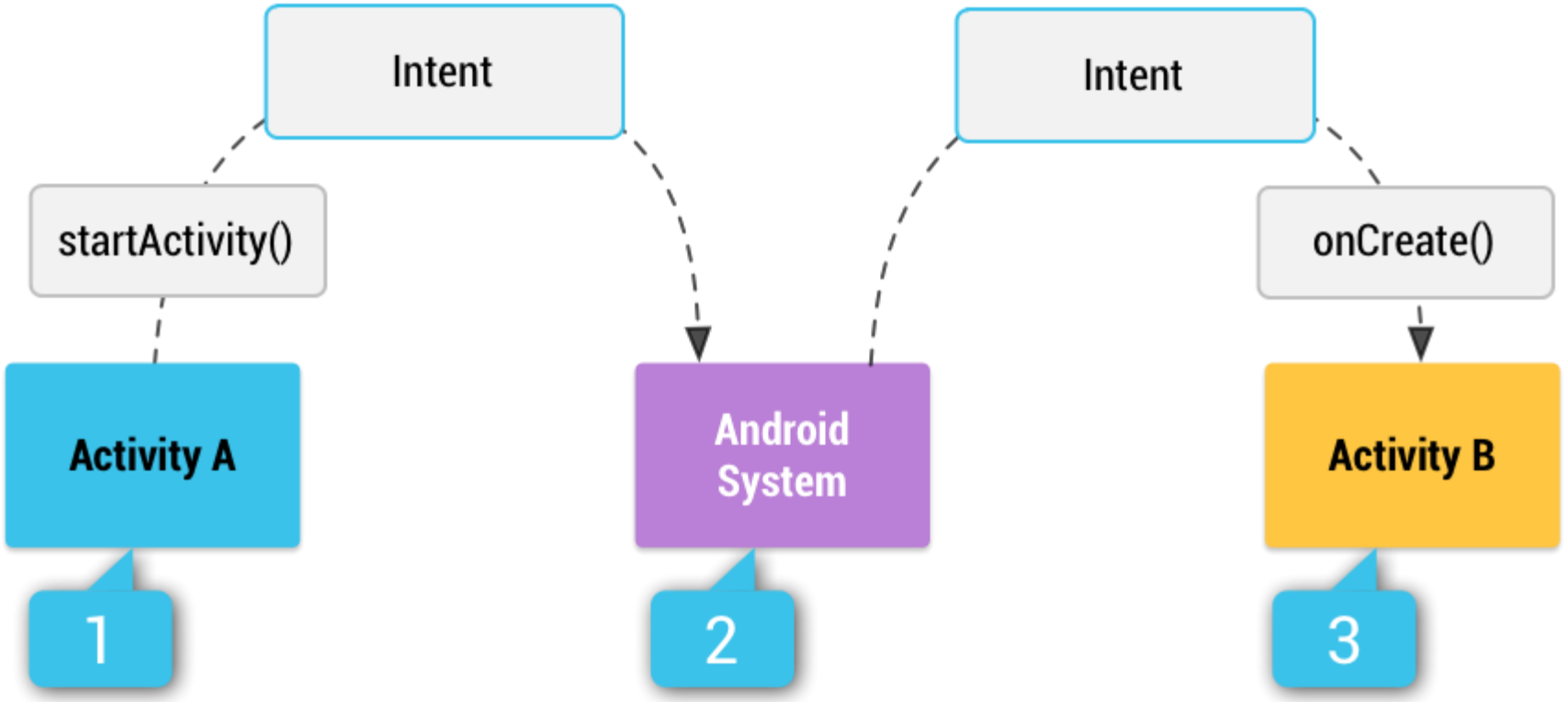
Android Intents

What are intents?

- Intents are asynchronous messages which allow Android components to request functionality from other components of the Android system.
- Intents can be used to signal to the Android system that a certain event has occurred.
- Other components in Android can register to this event via an intent filter.

Android Intents



Android Intents

Intent Types

- Explicit.

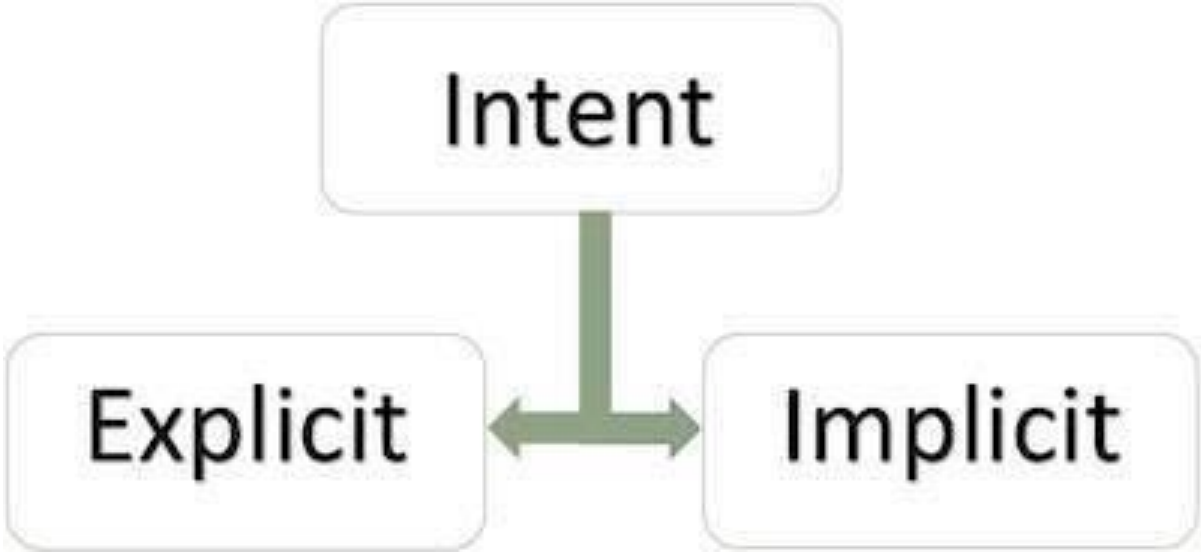
  - specify the component to start by name

  - the class name of the activity or service you want to start.

- Implicit.

  - do not name a specific component, but instead declare a general action to perform, which allows a component from another app to handle it.

Types of Intent



Explicit Intents

- These intents designate the target component by its name and they are typically used for application-internal messages - such as an activity starting a service or launching a another activity.

Explicit Intents

- Depending on the data included in the Intent object the Android system runs an receiver determination and determine the components which are responsible for the data described in the Intent object.
- The following code demonstrates how you can start another activity via an intent.

```
Intent i = new Intent(this, ActivityTwo.class);
startActivity(i);
```

Explicit Intents

The constructor used here takes two parameters

- Context as its first parameter (this is used because the Activity class is a subclass of Context)
- *Context in Android is an interface to global information about an application environment. This is an abstract class whose implementation is provided by the Android system. It allows access to application-specific resources and classes, as well as up-calls for application-level operations such as launching activities, broadcasting and receiving intents, etc.*
- The Class of the app component to which the system should deliver the Intent (in this case, the activity that should be started)

Implicit Intents

- These intents do not name a target and the field for the component name is left blank.
- Implicit intents are often used to activate components in other applications.

```
Intent read1=new Intent();
read1.setAction(Intent.ACTION_VIEW);
read1.setData(Uri.parse("http://google.co.in"));
startActivity(read1);
```

Intent Extras

```
Intent i= new Intent(this, Second.class);
i.putExtra("name", "Manish");
startActivity(i);
```

```
Intent i1=getIntent();
String s=i1.getStringExtra("name");
tv.setText(s);
```

Intent Extras

activity 1

```
    Intent pass = new Intent(this, Second.class);
    Bundle extras = new Bundle();
    extras.putString("nume", NUME_VAL);
    extras.putString("prenume", PRENUME_VAL);
    pass.putExtras(extras);
    startActivity(pass);
```

activity 2

```
public void onCreate(Bundle savedInstanceState)
{ super.onCreate(savedInstanceState);
    Bundle data = getIntent().getExtras();
    String name = data.getStringExtra("nume");
    String prename = data.getStringExtra("prenume");
  }
```

Pending Intent

- A **PendingIntent** is an object that wraps up an intent object and it specifies an action to be taken place in future.
- In other words, it lets us pass a future Intent to another application and allow that application to execute that Intent as if it had the same permissions as our application, whether or not our application is still around when the Intent is eventually invoked.
- The foreign app that receives the PendingIntent, doesn't know the content of Intent which is wrapped by PendingIntent.
- The mission of foreign app is to send back the intent to owner when some conditions are met (For example: alarm with schedule, or notification with click...). The conditions are given by owner but processed by foreign app (For example: alarm, notification).

Pending Intent

- A PendingIntent is generally used in cases where an **AlarmManager** needs to be executed or for **Notifications**(that we'll implement later in this tutorial).
- A PendingIntent provides a means for applications to work, even after their process exits.