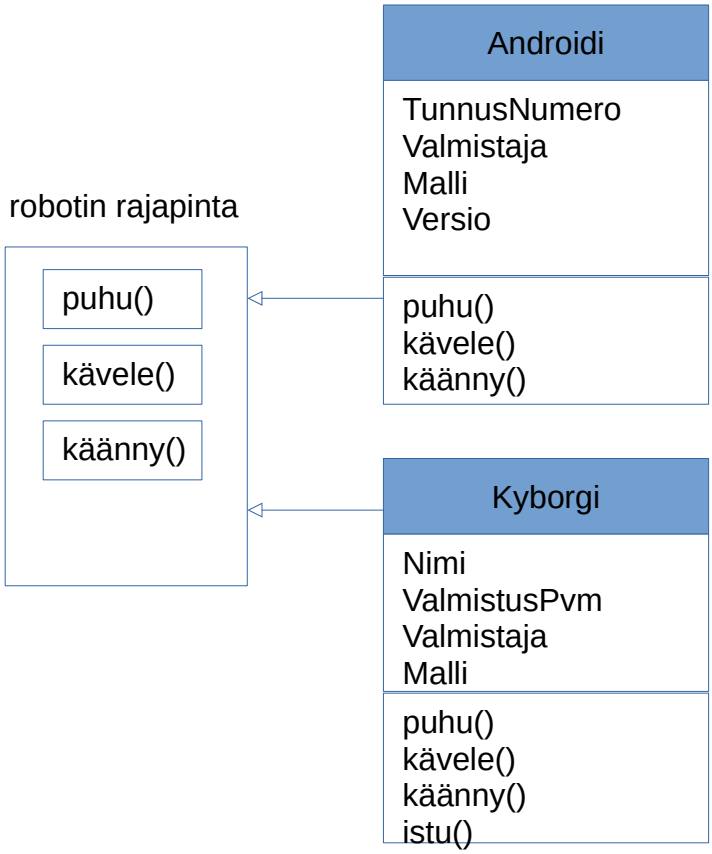
Rajapinta

Rajapinta (interface) on tarkoitettu keinoksi taata, että halutut metodit on toteutettu rajapintaa käyttävässä luokassa. Esimerkiksi voisi olla vaikkapa robotin rajapinta, jossa on kaikki se mitä robotin pitää osata tehdä. Toisin sanoen kaikki ne metodit, mitkä pitää toteuttaa tätä rajapintaa käyttävissä luokissa.

Rajapinta kertoo siis, mitkä ovat ne metodit ja ominaisuudet, joita käyttäen sovellusta tai ohjelmistokirjastoa käytetään. Rajapinta on siis todella tärkeä. Rajapinnassa olevien metodien ja ominaisuuksien avulla muut käyttävät sovellutustasi tai ohjelmistokirjastoasi. Rajapintaa siis ei kannata muuttaa esimerkiksi muuttamalla tai poistamalla rajapinnassa olevia metodeja tai ominaisuuksia. Lisää metodeja ja ominaisuuksia toki voi tehdä. Seuraavassa tarkastelemme, miten rajapintoja voi tehdä sovellutuksiin. Sellainen pitää olla jokaisessa sovellutuksessa tai ohjelmistokirjastossa.

Alla olevassa kuvassa on rajapinta robotista, jossa on metodit puhu(), kävele(), käänny(). Androidi ja Kyborgi luokat puolestaan käyttävät tätä rajapintaa. Sen vuoksi niissä on oltava toteutukset (ohjelmakoodi) siitä, mitä nämä metodit tekevät. Interface:ssä eli rajapinnassa itse toteutusta ei ole – interface:ssa vain kerrotaan, mitä pitää toteuttaa. Rajapintaa käyttävissä eri luokissa voi metodin toteutus olla erilainen. Androidi voisi siis kävellä eri tavalla kuin Kyborgi. Myös puhumisen ja kääntymisen voisi toteuttaa erilailla Androidille ja Kyborgille. Mutta kaikki rajapinnassa oleva täytyy toteuttaa. Tämä on rajapinnan idea.



Kuva 1 Rajapinta.

Yllä olevassa kuvassa on käytetty UML luokkakaaviota rajapinnan esittämiseen. Kuvassa 1 on esitetty myös, mitkä luokat käyttävät rajapintaa.

Ohjelmakoodina yllä oleva kuva näyttäisi seuraavanlaiselta. Huomaa, miten rajapinnan nimessä on käytetty isoa I -kirjainta, kertomaan rajapinta (IRobotti). Rajapinnat kannattaa nimetä tällä tavalla.

```
public interface IRobotti {
      public String puhu();
      public void kavele(Kohdepiste kordinaatti);
      public void kaanny(Astetta numero);
}
```

```
public class Androidi implements IRobotti {
	private int tunnusNumero;
	private String valmistaja;
	private String malli;
	private String versio;
	public String puhu() {
		return ”Hei, mitä kuuluu?”;
	}
	public void kavele(Kohdepiste koordinaatti) {
		// koodia, jolla voidaan kävellä...
	}
	public void kaanny(Astetta numero) {
		// koodia, jolla voidaan kääntyä….
	}
}
```

Jos puolestaan rajapinnan eli interface:n kaikkia metodeja ei halutakaan toteuttaa rajapintaa käyttävässä luokassa, on nämä metodit merkittävä `abstract`:eiksi interface:ssä. On siis tosiaankin olemassa keino välttää toteuttamasta rajapinnan kaikkia metodeja sitä käyttävässä sovellutuksessa. Alla olevassa esimerkkikoodissa esitetään, miten se onnistuu:

```
public interface IPeto{

	public abstract String miau();
	public abstract void tassuttele(Kohdepiste koordinaatti);
	public abstract void hyokkaa(Kohde elain);
	public abstract void syo(Kohde ruoka);
}

public class Kissa implements IPeto {
	private String nimi;
	private String rotu;
	private Date syntymaAika;

	public String miau() {
		return ”Miauu...”;
	}
	public void tassuttele(Kohdepiste koordinaatti) {
		// koodia, jolla voidaan tassutella...
	}
	public void hyokkaa(Kohde elain) {
		// koodia, jolla voidaan hyökätä….
	}
}
```

IPeto nimistä rajapintaa käyttävään Kissa -luokkaan ei siis tarvitsekaan tehdä kaikkia rajapinnassa olevia metodeja. Huomaa, miten siitä puuttuu `syo` -metodi!