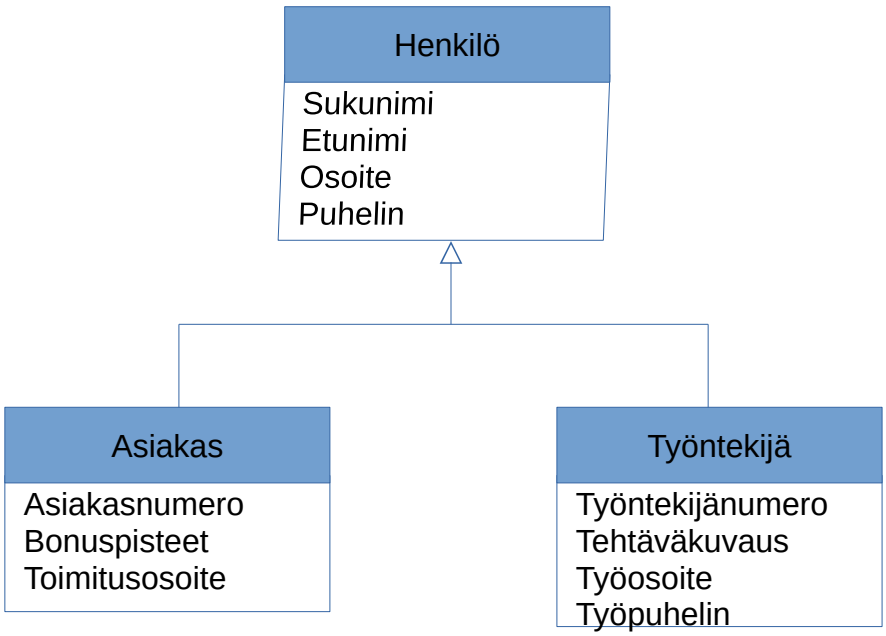
Periytyvyys

Periytyvyys tarkoittaa nimensä mukaan ominaisuuksia, jotka peritään vanhemmilta eli yliluokalta lapsille eli aliluokille. Esimerkiksi lapsi perii vanhempiensa ominaisuuksia. Tämä myös on tuotu olio-ohjelmointiin. Esimerkiksi jos yliluokka on Henkilö -luokka ja sen aliluokkia ovat Asiakas ja Työntekijä -luokat, peritytyvät Henkilö luokan omaisuudet ja myös sen metodit aliluokille. Tällaista periytyvyyttä voidaan kuvata niin sanotun luokkakaavion avulla:



Kuva 1: Periytyvyys Henkilö luokasta Asiakas- ja Työnteijä luokkiin.

Tässä Asiakas ja Työntekijä -luokat perivät Henkilö -luokasta sukunimen, etunimen, osoitteen ja puhelimen. Java -kielellä toteutettuna ohjelma näyttäisi alla olevalta ohjelmalta. Huomaa, että Asiakas ja Tyontekija luokissa periytyminen saadaan tehtyä `extends` avainsanalla. Extends avainsanan jälkeen on luokka, josta halutaan periyttää ominaisuuksia ja metodeja:

```
package henkilot;

public class Henkilo {
	private String sukunimi;
	private String etunimi;
	private String osoite;
	private String puhelin;
}

package henkilot:
```

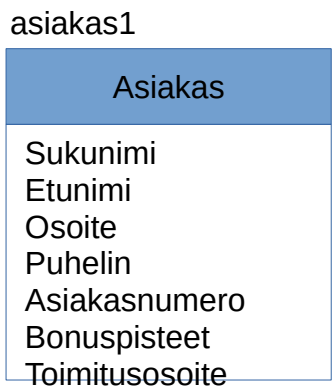
```
public class Asiakas extends Henkilo {
	private int asiakasnumero;
	private int bonuspisteet;
	private String toimitusosoite;
}
```

```
package henkilot;

public class Tyontekija extends Henkilo {
	private int tyontekijanumero;
	private String tehtavakuvaus;
	private String tyoosoite;
	private String tyopuhelin;
}
```

Nyt siis pystymme periyttämään ominaisuuksia aliluokille yliluokasta. Koska oletuskonstruktori on jo olemassa luokissamme, pystymme esimerkiksi luomaan Asiakas luokastamme olion. Tämä Asiakas olio voitaisiin luoda vaikkapa toisesta luokasta alla olevaan tapaan:

```
package henkilot;

public class SovellusOhjelma {
	public static void main(String[] args) {
		Asiakas asiakas1 = new Asiakas();
	}
}
```

Yllä olevassa ohjelmakoodissa on luotu Asiakas -luokasta asiakas1 niminen olio. Tähän asiakas1 olioon voi tallentaa ohjelman ajon ajaksi ominaisuuksina asiakasnumeron, bonuspisteet ja toimitusosoitteen. Mutta sillä on myös sukunimi, etunimi, osoite ja puhelin, jotka siis ominaisuuksina periytyvät yliluokasta. Tilannetta voisi siis kuvata alla olevan olion näköisenä tilanteena:



Kuva 2. Asiakas -luokasta luotu asiakas1 -niminen olio.

Saimme nyt siis luotua yhden uuden olion sovelluksessamme. Haluamme kuitenkin pystyä luomaan olioita myös siten, että annamme oliollemme data-arvoja. Nythän asiakas1 oliolle on täysin tyhjä data-arvoista. Tämän vuoksi lisäämme seuraavaksi Henkilo-, Asiakas- ja Tyontekija luokkiimme oletus- ja parametrilliset muodostimet (konstruktorit).

```
package henkilot;

public class Henkilo {
	private String sukunimi;
	private String etunimi;
	private String osoite;
	private String puhelin;

	// Oletusmuodostin eli konstruktori:
	public Henkilo() {
	}

	// Parametrillinen muodostin:
	public Henkilo(String sukunimi, String etunimi, String osoite,
		String puhelin) {
		this.sukunimi = sukunimi;
		this.etunimi = etunimi;
		this.osoite = osoite;
		this.puhelin = puhelin;
	}
}

package henkilot:

public class Asiakas extends Henkilo {
	private int asiakasnumero;
	private int bonuspisteet;
	private String toimitusosoite;

	// Oletusmuodostin:
	public Asiakas() {
		super();
	}

	// Parametrillinen muodostin:
	public Asiakas(String sukunimi, String etunimi, String osoite,
		String puhelin, int asiakasnumero, int bonuspisteet,
		String toimitusosoite) {

		super(sukunimi, etunimi, osoite, puhelin);
		this.asiakasnumero = asiakasnumero;
		this.bonuspisteet = bonuspisteet;
		this.toimitusosoite = toimitusosoite;
	}

}
```

```
package henkilot;

public class Tyontekija extends Henkilo {
	private int tyontekijanumero;
	private String tehtavakuvaus;
	private String tyoosoite;
	private String tyopuhelin;

	// Oletuskonstruktori eli muodostin
	public Tyontekija() {
		super();
	}

	// Parametrillinen muodostin:
	public Tyontekija(String sukunimi, String etunimi, String osoite,
		String puhelin, int tyontekijanumero, String tyoosoite,
		String tyopuhelin) {

		super(sukunimi, etunimi, osoite, puhelin);
		this.tyontekijanumero = tyontekijanumero;
		this.tyoosoite = tyoosoite;
		this.tyopuhelin = tyopuhelin;
	}

}
```

Huomaa, miten yllä olevassa ohjelmassa on käytetty `super()`:ia osoittamaan yliluokan oletuskonstruktoriin. Super():illa saadaan ajettua käyntiin (kutsuttua) yliluokan oletuskonstruktori.

Nyt pystymme luomaan asiakas2 olion, johon saamme ohjelman ajon ajaksi talletettua keskusmuistiin asiakkaan kaikki tiedot:

```
package henkilot;

public class SovellusOhjelma {
	public static void main(String[] args) {
		Asiakas asiakas1 = new Asiakas();
		Asiakas asiakas2 = new Asiakas(”Poppanen”, ”Maija”, ”Velhokuja 1”,
			”011-98765412”, 1, 0, ”Taikurikatu 10”);
	}
}
```

Nyt on saatu luotua uusi asiakas2 niminen olio, jonka luonnin yhteydessä saatiin talletettua siihen halutut data-arvot ajon ajaksi. Tilannetta voisi asiakas2 olion osalta kuvata alla olevalla kuvalla:

asiakas2

Asiakas
Sukunimi: ”Poppanen” Etunimi: ”Maija” Osoite: ”Velhokuja 1” Puhelin: ”011-98765412” Asiakasnumero: 1 Bonuspisteet: 0 Toimitusosoite: ”Taikurikatu 10”

Kuva 3. Asiakas2 olion sisältämät dataarvot.

Tähän mennessä olemme vain tehneet erilaisia ominaisuuksia ja muodostimia luokkiimme. Niiden avulla olemme pystyneet tallentamaan data-arvoja olioon ja muodostimilla luomaan olioita halutulla tavalla. Seuraavaksi lisäämme luokkiimme metodeja. Tämän jälkeen luokistamme luomamme oliot pystyvät myös tekemään ohjelmoimiamme asioita. Myös metodeja voidaan periyttää aivan kuten ominaisuuksiakin. Jos siis teemme vaikkapa toString metodin Henkilo -luokkaan, on se käytettävissä myös Asiakas -luokassa tehdyssä oliossa. Tällä toString metodilla haluamme pystyä palauttamaan kaikki Henkilo luokassa olevat ominaisuudet. Itseasiassa haluamme samanlaisen metodin kaikkiin Asiakas ja Työntekijä luokkiin myös. Niissäkin on omia ominaisuuksia, joiden data-arvot haluamme tietää kätevästi kaikki kerralla. Näiden muutosten jälkeen ohjelmamme näyttää seuraavanlaiselta. Onneksi toString -metodin voi helposti generoida Eclipsessä valisemalla siellä: Source → Generate toString()... :

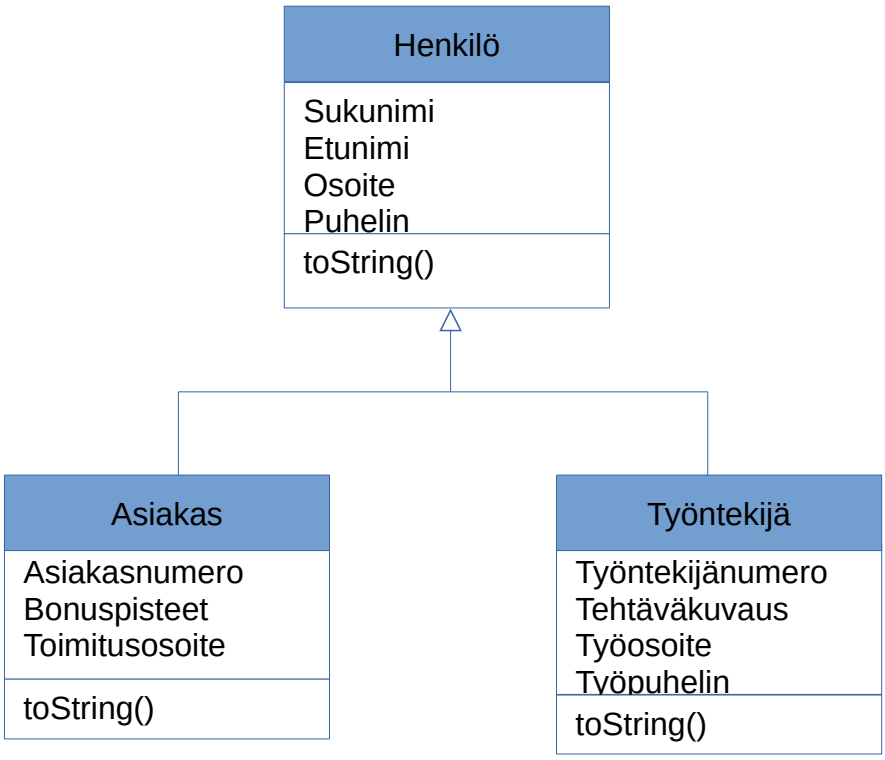
```
package henkilot;

public class Henkilo {
	private String sukunimi;
	private String etunimi;
	private String osoite;
	private String puhelin;

	// Oletus muodostin:
	public Henkilo() {
	}

	// Parametrillinen muodostin:
	public Henkilo(String sukunimi, String etunimi, String osoite,
		String puhelin) {
		this.sukunimi = sukunimi;
		this.etunimi = etunimi;
		this.osoite = osoite;
		this.puhelin = puhelin;
	}

	@Override
	public String toString() {
		return "Henkilo [sukunimi=" + sukunimi + ", etunimi=" + etunimi + ",
			osoite=" + osoite + ", puhelin=" + puhelin
	}
}
```

```
package henkilot:

public class Asiakas extends Henkilo {
	private int asiakasnumero;
	private int bonuspisteet;
	private String toimitusosoite;

	// Oletus muodostin:
	public Asiakas() {
		super();
	}

	// Parametrillinen muodostin:
	public Asiakas(String sukunimi, String etunimi, String osoite,
		String puhelin, int asiakasnumero, int bonuspisteet,
		String toimitusosoite) {

		super(sukunimi, etunimi, osoite, puhelin);
		this.asiakasnumero = asiakasnumero;
		this.bonuspisteet = bonuspisteet;
		this.toimitusosoite = toimitusosoite;
	}

	@Override
	public String toString() {
		return "Asiakas [asiakasnumero=" + asiakasnumero + ", bonuspisteet="
			+ bonuspisteet + ", toimitusosoite="
			+ toimitusosoite + ", toString()=" + super.toString() + "]";
	}
}
```

```
package henkilot;

public class Tyontekija extends Henkilo {
	private int tyontekijanumero;
	private String tehtavakuvaus;
	private String tyoosoite;
	private String tyopuhelin;

	public Tyontekija() {
		super();
	}

	// Parametrillinen muodostin:
	public Tyontekija(String sukunimi, String etunimi, String osoite,
		String puhelin, int tyontekijanumero, String tyoosoite,
		String tyopuhelin) {

		super(sukunimi, etunimi, osoite, puhelin);
		this.tyontekijanumero = tyontekijanumero;
		this.tyoosoite = tyoosoite;
		this.tyopuhelin = tyopuhelin;
	}

	@Override
	public String toString() {
		return "Tyontekija [tyontekijanumero=" + tyontekijanumero + ",
			tehtavakuvaus=" + tehtavakuvaus + ", tyoosoite="
			+ tyoosoite + ", tyopuhelin=" + tyopuhelin + ", toString()=" +
			super.toString() + "]";
	}

}
```

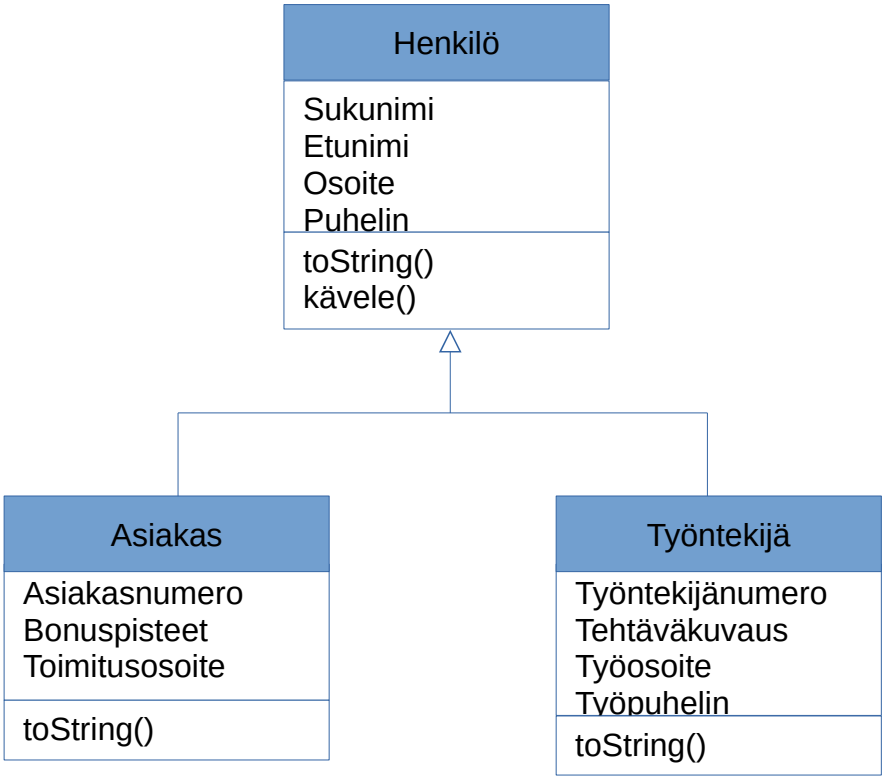
Seuraavaksi tähän esimerkki sovellukseen voisi lisätä uusina metodeina niin sanotut setterit ja getterit. Nekin saa generoitua samaan tapaan kuin edellä toString. Eli Eclipsen Source → Generate Getters and Setters. Niiden avulla voi nyt asettaa haluamansa data-arvot luotuihin olioihin ja palauttaa (”hakea”) nämä arvot halutessaan. Nämä muutokset löytyvät nyt zipattuna tiedostona Moodlen linkistä.

Nyt kun Henkilö, Asiakas ja Työntekijä luokissamme on myös jotain mitä ne osaavat tehdä, voisimme kuvata ne seuraavalla kuvalla alla:



Kuva 4. Periytyvyys Henkilö luokasta Asiakas ja Työntekijä luokkiin.

Voisimme tietysti lisätä muutakin toiminnallisuutta, jota esimerkiksi Henkilö luokka osaa tehdä - esimerkiksi kävellä. Tämän muutoksen jälkeen tilanne näyttäisi luokkakaaviossamme seuraavalta:



Kuva 5. Periytyvyys Henkilö luokasta Asiakas ja Työntekijä luokkiin.

Nyt myös Asiakas ja Työntekijä luokat, Henkilö luokan lisäksi, pystyvät ”kävelemään”. Esimerkiksi:

```
Tyontekija tyontekija1 = new Tyontekija();
System.out.println(tyontekija1.kavele());
```

Jos Henkiö luokkaan lisättäisiin kyseinen metodi:

```
public String kavele() {
      return ”Kävelen nyt eteenpäin….”;
}
```