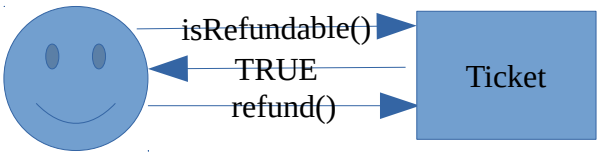
Liides ja abstraktne klass

Lennufirma palkas teid arendama uut piletimüügisüsteemi. Teie esimeseks eesmärgiks on programmeerida piletite klassid. Lennufirma pakub nelja piletiklassi, millele kehtivad erinevad tingimused:

	Plain	**Common**	**Premium**	**Enjoy**
Pileti tagastamine	ei	ei	jah	jah
Nime muutmine	Kuni 30 päeva enne lenu	Kuni 20 päeva enne lendu	Kuni 10 päeva enne lendu	Jah, enne lendu
Lennuaja muutm.	100 EUR	50 EUR	tasuta	tasuta
Käsipagas	8 kg	10 kg	10 kg	12 kg
Äraantav pagas	-	20 kg	40 kg	40 kg
Äriklassi check-in	ei	ei	ei	jah
Toitlustus	10 EUR	10 EUR	tasuta	tasuta

Ühelt poolt peavad need klassid tagastama informatsiooni pakutavate võimaluste kohta (nt meetod isRefundable() võiks tagastada, kas piletit saab lennufirmale tagastada või mitte). Teisalt peavad need klassid haldama ka piletiga seotud tegevusi. Näiteks kui kasutaja pilet on lennufirmale tagastatav, siis peab seal olema meetod refund(). Kui see võimalus puudub, ei tohiks ka antud meetodit olla. Võimalik kasutusjuht:



Kuivõrd erinevaid tingimusi on iga piletitüübi kohta mitmeid, oleks seda süsteemi üsna kentsakas realiseerida puhtalt pärimist ja polümorfismi kasutades. Seetõttu kasutame mitmete omaduste defineerimiseks liideseid.

Ülesanne 1 – ettevalmistus

Looge klass Praks13, sinna main meetod, kust hakkate piletimüügisüsteemi kasutama.

Ülesanne 2

Looge paketti airways abstraktne klass Ticket. Looge sinna konstruktor, mis võtab argumendiks reisija nime ning `Calendar` tüüpi argumendina lennu väljumise aja. Salvestage need klassi väljadele. Milline peaks olema väljade nähtavus (arvestage, et hakkame klassi laiendama)? Looge mõlema omaduse kohta getter meetod.

Ülesanne 3

Looge klassid iga piletitüübi jaoks laiendades klassi Ticket.

Ülesanne 4

Looge pileti tagastamise kohta informatiivne meetod: abstraktses klassis Ticket defineerige meetod isRefundable(), mis tagastab kas true või false. Konkreetsetes klassides kirjutage see meetod vajadusel üle.

Ülesanne 5

Looge liides Refundable ning defineerige seal meetod public void refund()

Ülesanne 6

Klassid, kus pileti tagastamine on võimalik, peaksid realiseerima liidest Refundable.

refund() meetodi sisuks võib olla konsooli output, nt „Pilet tagastatud”.

Ülesanne 7

Looge Praks13 klassis igast piletitüübist uus objekt ja kutsuge välja isRefundable() meetodid ja tagastage pilet, kui see on tagastatav. Calendar objekti loomise näide:

```
Calendar c = new GregorianCalendar();
c.set(2015, 0, 15);
```

Üldprintsiibi järgi peaksite pileti tagastamiseks kirjutama sellise koodi:

```
Ticket t = new PremiumTicket("Martin", c2);

if (t.isRefundable()) {
	t.refund();
}
```

Ent selline kood ei tööta, kuna Java käsitleb objekti t Ticket-tüüpi objektina. Ticket klassis aga ei ole olemas meetodit refund(), seega see kood ei kompileeru. Seetõttu tuleb muutuja t **teisendada** Refundable objektiks – Java kompilaator teab, et sel objektil on refund() meetod kindlasti olemas.

```
if (t.isRefundable()) {
	Refundable r = (Refundable)t;
	r.refund();
}
```

Alternatiivne võimalus on see realiseerida ilma isRefundable() meetodita (eeldame, et kõiki Refundable liidest laiendavaid pileteid saab alati tagastada):

```
if (t2 instanceof Refundable) {
	Refundable r = (Refundable)t;
	r.refund();
}
```

Ülesanne 8

Järgmiseks realiseerige nime muutmise funktsionaalsus. Nagu tabelist näeme, on igal piletitüübil oma tingimused nime muutmiseks. Mõistlik oleks luua kaks meetodit: üks, mis ütleb, kas nime saab muuta (kas tingimused on täidetud) ja nime muutmiseks. Esimene meetod oleks kõige otstarbekam luua abstraktsena klassis Ticket – palun tehke seda. Tuletage meelde, et abstraktsel meetodil ei ole sisu (sarnaselt liideses defineeritud meetodile) ja vajab definitsioonis **abstract** keywordi.

Ülesanne 9

Realiseerige eelmises ülesandes loodud meetod kõigis klassides. Vihje: Ticket klassi laiendavates klassides saate Eclipse abil meetodi skeletid automaatselt genereerida (Add unimplemented methods). Kuupäevi saab võrrelda näiteks nii (aeg on Calendar tüüpi objekt, mille defineerisite Ticket klassis):

```
Calendar c2 = new GregorianCalendar();
// lisame tänasele päevale 30 päeva
c2.add(Calendar.DAY_OF_YEAR, 30);

if (c2.before(aeg)) {
	// piletit saab vahetada
} else {
	// piletit ei saa vahetada
}
```

Ülesanne 10

Looge klassi Ticket konkreetne meetod nime muutmiseks. Kontrollige, kas tingimused on nime muutmiseks olemas (kasutades enne loodud meetodit) – kui kõik on korras, siis muutke klassiväljal nimi ja trükkige konsooli „Nimi muudetud”, vastasel juhul „Nime muutmine pole enam lubatud”.

Kutsuge funktsionaalsus iga piletitüübi jaoks välja Praks13 klassist ja veenduge, et see töötab mõlemal juhul:

```
Calendar c3 = new GregorianCalendar();
c3.add(Calendar.DAY_OF_YEAR, 5); // lisame viis päeva alates tänasest

Ticket t3 = new PremiumTicket("John", c3);
t3.changeName("Foo bar"); // nime muutmine ei tohi olla lubatud

Calendar c4 = new GregorianCalendar();
c4.add(Calendar.DAY_OF_YEAR, 15); // lisame viisteist päeva tänasest

Ticket t4 = new PremiumTicket("Mary", c4);
t4.changeName("Java Guru"); // nime muutmine õnnestub
```

Ülesanne 11

Realiseerige äraantava pagasi funktsionaalsus liidese abil, sarnaselt pileti tagastamise funktsionaalsusega.

Kutsuge funktsionaalsus iga piletitüübi jaoks välja Praks13 klassist ja veenduge, et see töötab.

Ülesanne 12

Realiseerige kaasavõetava pagasi funktsionaalsus abstraktse meetodi abil sarnaselt nime muutmisega.

Kutsuge funktsionaalsus iga piletitüübi jaoks välja Praks13 klassist ja veenduge, et see töötab.

Boonus

Kõik ülesanded 1-10 täielikult lahendanud tudengid saavad 2 boonuspunkti. Ülesanne tuleb zip failina laadida Moodlesse „Teema 13” all oleva esitamise lingi kaudu hiljemalt 02. detsembri varahommikul kell 1:59.