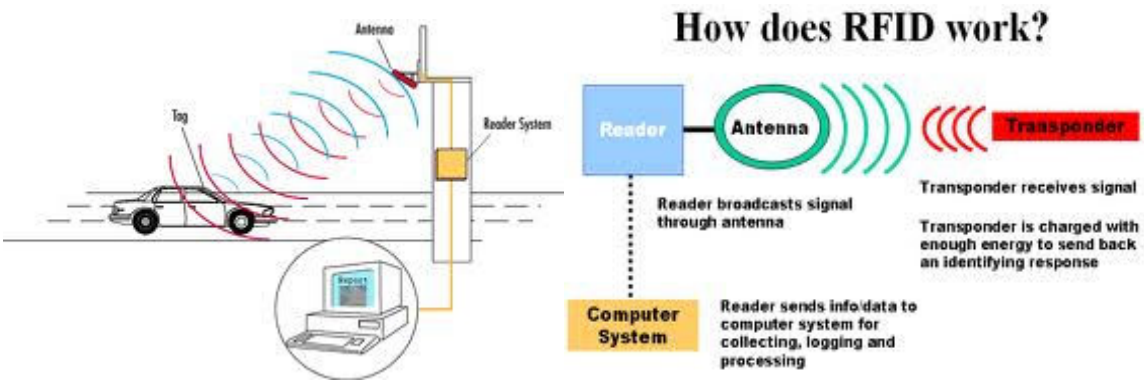
Passersystem

RFID- tekniken (Radio Frequency ID) används i allt fler applikationer, från passersystem och biltullstranspondrar till varoretiketter.



Passersystemet består av en läsare (*Reader*) och en passerkort (*Transponder*), där kommunikationen sker med hjälp av induktion. När taggen kommer inom läsarantennens fält laddas taggens kondensator, och taggen kan sända informationen (transponderns unika ID-kod för read-only chip) till läsaren. Till skillnad mot streckkoden behöver taggen inte vara synlig för sändare/mottagarenheten, men den får heller inte vara för långt borta. Dagens läsavstånd är mellan två centimeter och 5 meter.

Tekniken passar mycket väll för ett passersystem till en till exempel skidlift. Passerkorten innehåller RFID-tag som har ett unikt nummer bestående av 10 siffror.

När ett nytt passerkort säljs lagras RFID-numret i en datastruktur, vi kallar den accessträd.
Vid varje passage sker sökning i systemets i accessträd. Om det är ett giltigt RFID-nummer, godkänns passage annars inte.

Antalet skidåkare kan variera väldigt mycket från några få till flera hundra. Eftersom många nya skidkort kontinuerligt skall läggs in eller tas bort från systemet, krävs en mycket dynamisk datastruktur för att lagra accessinformationen i. Eftersom sökning är mycket effektiv i binära sökträd, väljer vi **ett träd liknande datastruktur, dvs en trädbaserad ”symbol tabell”**.

Passerkortet finns färdig definierad i klassen Card.java. Ett Card-objekt består utöver id-numret också av information om giltighetsdatum *(validTo)*. Detta för att vi vill att systemet automatiskt skall ta bort passerkorten från accesslistan. I

klassen Card finns det en main() där du kan se hur Calendar och Date klassen används då vi behöver programmera ”tider”.

Den andra klassen är Node.java (där får du ändra om du så tycker). En Nod består av ett Card-objekt, en nyckel som är RFID kortens unika nummer (av typ String) samt länkar till två andra noder i trädet.

Accessträdet är från början en tomt träd som endast består av en ”tom” Nod. Allt eftersom skidkort säljs, läggs det nya noder till trädet. När kortet blir ogiltigt tas bort noden med respektive Card från trädet.

För att dessa operationer skall kunna utföras måste du först implementera metoderna från klassen AccessTree.java. Se implementationen av trädbaserade symbol-tabeller i din online bok samt i filen du har fått.

a) Skriv ett testprogram där du simulerar hur systemet fungerar.
Börja med att anta att du har sålt 500 kort. Låt de åka. Börja med att lägga i kön först 100 åkare. Resten kan läggas in en efter en efter en slumpad tid.
Lägg korten i liftkön med en mellan rum mellan (1-3 sekunder) men också slumpa (t.ex. mellan 1-5 kort) antal kort som läggs i kön innan en åkare tas bort från kön. Från liftkön plockas ut ett kort i taget med 1 sekund mellanrum.
Börja med:

- Skapa Accessträdet (datastrukturen)

- ”Sälj skidkort”, dvs skapa Card-objekt och lägg den i trädet. Försök automatisera skapandet av passerkorten (dvs, generera rfid-nummer i en metod samt slumpa antal sekunder som kortet är giltig).

- Skapa Kö-datastrukturen och lägg 100 kort i den.

- ”Ta bort” Card-objekt från access trädet, då korten har blivit ogiltiga. Kan du hitta en lämpligt lösning för att lösa detta?

- När kortet ”passerar” ska ett meddelande som ”Access accepted” eller ”Access denied ” skrivas.

b) Om din sökningsalgoritm verkligen är O(logN) borde dina sökningstider du fick fram genom mätning stämma med teorin.
Gör ett enkelt program där du först skapar 1000 kort och lägger dem i ett access träd. Sök 10 gånger efter 10 olika kortnummer. Mät tiden som det tar för sökningen. Läs tiden från datorns interna klocka med metoden

```
long timeNow = System.currentTimeMillis();
```

både innan du startar sökningen och efter det.
Spara söktiden för varje sökning. Beräkna genomsnittstiden för alla 10 sökningarna.
Gör samma sak för ett träd som innehåller 5 000 – samt 10 000 kort.
Växer exekveringstiden för din metod logaritmiskt eller inte? Varför borde den vara logaritmiskt?

Skriftlig rapport

Din rapport skall innehålla en beskrivning om hur du har byggt din simulering och hur den lever upp till kravspecifikationen. Rapporten ska även innehålla vad du tycker att du har lyckats med och vad du inte lyckats med vad gäller simuleringsprogrammet.
Du skall beskriva och förklara det resultat som du har fått från dina mätningar som du har gjort.
Vad tror du? Är din simulering bra nog för att kunna ge rekommendationer om systemet håller eller inte måttet. Varför? Varför inte?
Rapporten behöver inte vara längre än 2 A4 sidor.

Rapporten och koden skall skickas till mig. Alla filer skall vara packade i en zip eller rar fil som har era namn t.ex. **AntonOchAdam.zip**