PH Bicycle Garage 8000 - Designdokument

Gruppmedlemmar:
Mattias Nordahl (dt07mn0@student.lth.se)
Hannes Nevalainen (dt07hn2@student.lth.se)
Daniel Olofsson (dt07do1@student.lth.se)
Fredrik Andersson (dt07fa5@student.lth.se)
Emil Einarsson (dt07ee3@student.lth.se)
Sandra Nilsson (dt07sn8@student.lth.se)

Gruppnr: 18

Kursansvarig: Martin Höst

Dokumentansvarig: Fredrik Andersson
Version: 1.3

2008-05-18

Innehåll

1 Ändringshistorik

Version	Datum	Ansvarig	Beskrivning
1.0	2008-04-28	F.A.	Satt i baseline
1.1	2008-05-05	F.A.	**User:** *getPinCode()*, *setPinCode()* samt *getBicycles()* har lagts till, personalNumber ändrad till *long* istället för *int*. **DBM:** *getUserFromPersonalNumberAndPinCombination()* har lagts till. **BGM:** barCode är inte längre en parameter i *newBicycle()* eller *editBicycle()*.
1.2	2008-05-14	F.A.	**FM:** Tar numera en *Logger* i konstruktorn, konstruktorn kan kasta ett IOException, två nya metoder *removeBicycle()* samt *removeUser()*. **DBM:** Tar numera en *Logger* i konstruktorn. **BGM:** Tar numera en *Logger* i konstruktorn, två nya metoder *save()* tar *Bicycle* eller *User* objekt, ny metod *assignNewPinCode()*.
1.3	2008-05-18	F.A.	**BGM:** *newUser()* samt *newBicycle()* returnerar objektet som skapades, *setBicycycleLocation()* tillagd, *editBiycle()* har fått uppdaterade parametrar. **BGG:** *update()* tillagd.

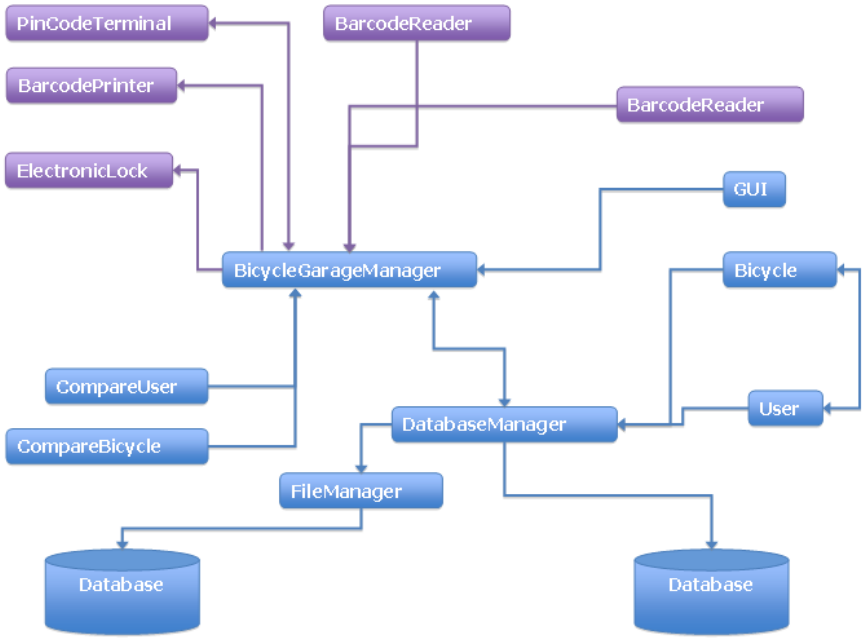
2 Introduktion

Systemet är uppdelat i ett flertal olika klasser där vi har försökt placera varje större funktion i en egen klass. En schematisk bild över systemet kan ses i Figur 1. Den klassen som ligger närmast användaren är BicycleGarageGUI vilken tar emot alla kommandon och skickar vidare dessa till BicycleGarageManager som är det centrala i prorgammet. BicycleGarageManager tar hand om alla kommandona som kommer från GUI och alla hårdvarudrivrutiner samt ser till att samordna arbetet mellan dessa. Denna designen gör det möjligt att enkelt byta ut användargränssnittet mot något annat om så skulle önskas, utan att behöva implementera om funktionalitet i programmet.

Till sin hjälp har BicycleGarageManager klassen DatabaseManager som sköter en stor del av de operationer som utförs på stora grupper av data, t ex sortering, sökning och inläsning av data. DatabaseManager tar hjälp av klassen FileManager för skrivning och inläsning från disk, men det är DatabaseManager som har i uppgift att hantera datan och se till att den är åtkomlig för klassen BicycleGarageManager.

Ytterliggare klasser är User och Bicycle som representerar användare respektive cyklar i systemet. För att underlätta inbördes jämförelse mellan dessa finns dessutom två komparatorklasser CompareUser och CompareBicycle.

I detta dokumentet presenteras en detaljerad beskrivning över alla publika metoder i klasserna. Till de flesta finns även en beskrivning över vad de ska göra. Dessa beskrivningar måste följas vid implementationen för att göra det möjligt för flera personer att utveckla de olika klasserna parallellt med varandra. En del enklare metoder som har namnen set<Attribut> och get<Attribut> saknar detaljerad beskrivning, det enda som ska ske i dessa metoder är att tilldela respektive returnera värdet på ett privat attribut i klassen.



Figur 1: Schematisk bild över designen

3 Klasser

3.1 Bicycle

3.1.1 Beskrivning

Ansvarig för klassen: Daniel Olofsson och Sandra Nilsson

Denna klass representerar ett cykelobjekt. Varje cykel som finns i systemet kommer att vara representerad som ett objekt av denna klass. Varje instans av klassen innehåller en referens till den användare som den är kopplad till. Vidare innehåller den all ytterliggare nödvändig information som enligt krav 5.3.2.2 ska lagras för varje cykel. Det finns dessutom inbyggd funktionalitet för att hantera nödvändig loggning enligt krav 5.3.4.2. Observera att klassen måste ta hänsyn till krav 5.3.5.6.

3.1.2 Konstruktorer

```
/**
 * Denna konstruktor bör användas vid skapande av en ny Bicycle. Det
     kommer
 * automatiskt att skapas en ny barCode. Fälten lastEnter, lastExit,
 * dateRegister sätts automatiskt till rätt värde.
 *
 * @param description
 *             Beskrivning av cykeln
 * @param frameNumber
 *             Eventuellt ramnummer på cykeln
 * @param owner
 *             Ägare av cykeln
 * @param notes
 *             Anteckningar om cykeln
 */
public Bicycle(String description, String frameNumber, User owner,
    String notes)

/**
 * Denna konstruktor bör endast användas av FileManager vid
     återskapandet av
 * datan som finns sparad på disken.
 *
 * @param barCode
 * @param description
 * @param frameNumber
 * @param owner
 * @param notes
 * @param location
 * @param lastEnter
 * @param lastExit
 * @param dateRegister
 */
public Bicycle(int barCode, String description, String frameNumber, User
     owner, String notes, int location, Date lastEnter, Date lastExit,
    Date dateRegister)
```

3.1.3 Metoder

```
/**
 * Kontrollerar om cykeln befinner sig i garaget.
 *
 * @return true om cykeln är i garaget, annars false
 */
public boolean isInGarage()

public String getDescription()
```

```
public void setDescription(String description)
public String getFrameNumber()
public void setFrameNumber(String frameNumber)
public User getOwner()
public void setOwner(User owner)
public String getNotes()
public void setNotes(String notes)
public int getLocation()
public void setLocation(int location)
public Date getLastEnter()
public void setLastEnter(Date lastEnter)
public Date getLastExit()
public void setLastExit(Date lastExit)
public int getBarCode()
public Date getDateRegister()
```

3.2 User

3.2.1 Beskrivning

Ansvarig för klassen: Emil Einarsson

Denna klass representerar ett användarobjekt. Varje användare som finns i systemet kommer att vara representerad som ett objekt av denna klass. Varje instans av klassen innehåller en samling av referenser till de Bicycle objekt som kopplas ihop med den. Vidare innehåller den all ytterligare nödvändig information som enligt krav 5.3.2.1 ska lagras för varje användare. Det finns dessutom inbyggd funktionalitet för att hantera nödvändig loggning enligt krav 5.3.4.2. Observera att klassen inte behöver ta hänsyns till krav 5.3.1.9 utan det hanteras av de klasser som använder sig av denna klass.

3.2.2 Konstruktorer

```
/**
 * Denna konstruktor bör användas vid skapande av en ny User. De 6
     första
 * siffrorna i personnumret kombinerat med PIN-koden måste utgöra en
     unik
 * kombination för varje användare. Fälten dateRegister, lastVisist,
 * nbrEntries sätts automatiskt till rätt värde.
 *
 * @param personalNumber
 *              Personnumret på användaren
 * @param pin
 *              PIN-kod för användaren
 * @param name
 *              För- och efternamn på användaren
 * @param homePhone
 *              Telefon hem till användaren
 * @param mobilePhone
 *              Mobiltelefon för användaren
 * @param address
 *              Adress till användaren
```

```
     * @param notes
     *             Anteckningar om användaren
     */
    public User(long personalNumber, int pin, String name, String homePhone,
            String mobilePhone, String address, String notes)

    /**
     * Denna konstruktor bör endast användas av FileManager vid
         återskapandet av
     * datan som finns sparad på disken.
     *
     * @param personalNumber
     * @param pin
     * @param name
     * @param homePhone
     * @param mobilePhone
     * @param address
     * @param notes
     * @param dateRegister
     * @param lastVisit
     * @param nbrEntries
     * @param bicycleList
     */
    public User(long personalNumber, int pin, String name, String homePhone,
            String mobilePhone, String address, String notes, Date
            dateRegister, Date lastVisit, int nbrEntries, List<Bicycle>
            bicycleList)
```

3.2.3 Metoder

```
    /**
     * Kontrollerar om användaren har en viss PIN-kod.
     *
     * @param i
     *             PIN-koden som vill kontrolleras
     * @return true om i är användarens PIN-kod
     */
    public boolean hasPIN(int i)

    /**
     * Lägger till en Bicycle till användaren. En cykel som redan ägs av
     * användaren kan inte läggas till.
     *
     * @param b
     *             Den Bicycle som önkas tilldelas användaren
     * @return true om det lyckades, annars false
     */
    public boolean addBicycle(Bicycle b)

    /**
     * Tar bort en Bicycle från användaren.
     *
     * @param b
     *             Den Bicycle som önskas tas bort.
     * @return true om det lyckades, annars false
     */
    public boolean removeBicycle(Bicycle b)

    /**
     * Returnerar alla cyklar som användaren är kopplad till.
     *
     * @return en vektor med Bicycle objekt som användaren är kopplad till.
     */
    public Bicycle[] getBicycles()

    /**
     * Ökar användarens antal registrerade inträden i lokalen med 1.
     */
    public void increaseEntryCount()

    public long getPersonalNumber()

    public void setPersonalNumber(long personalNumber)
```

```
public int getPin()

public void setPin(int pin)

public String getName()

public void setName(String name)

public String getHomePhone()

public void setHomePhone(String homePhone)

public String getMobilePhone()

public void setMobilePhone(String mobilePhone)

public String getAddress()

public void setAddress(String address)

public String getNotes()

public void setNotes(String notes)

public Date getLastVisit()

public void setLastVisit(Date lastVisit)

public Date getDateRegister()

public int getNbrEntries()
```

3.3 FileManager

3.3.1 Beskrivning

Ansvarig för klassen: Fredrik Andersson

Detta är klassen som jobbar mot filerna. Programmet har ett eget format som den följer när data skrivs till disk. Denna klass har till uppgift att hantera skrivning och läsning på ett korrekt sätt. Den gör om Bicycle- och User-objekt så att de kan sparas på en fil på disk och kan även gå omvänt, från en fil på disk till Bicycle/User objekt. Innehåller även viss loggning av operationerna som utförs.

3.3.2 Konstruktorer

```
/**
 * Skapar en FileManager som arbetar mot valda filnamn.
 *
 * @param bicycleFileName
 *              Namnet på filen som kommer att innehålla data om Bicycle
 *              objekten
 * @param userFileName
 *              Namnet på filen som kommer att innehålla data om User
    objekten
 * @param log
 *              Logger objekt som kommer att användas av klassen vid
    loggning
 * @throws IOException
 *              Om det blev fel vid skapandet på filerna, eller om
    klassen
 *              inte har åtkomst till nödvändiga operationer på dessa.
 */
public FileManager(String bicycleFileName, String userFileName, Logger
    log)
```

3.3.3 Metoder

```
/**
 * Skriver ett valt Bicycle objekt till disken.
 *
 * @param b
 *             Bicycle objektet att spara
 * @return true om det lyckades, annars false
 */
public boolean saveBicycle(Bicycle b)

/**
 * Skriver ett valt User objekt till disken.
 *
 * @param u
 *             User objektet att spara
 * @return true om det lyckades, annars false
 */
public boolean saveUser(User u)

/**
 * Tar bort ett valt Bicycle objekt från disken.
 *
 * @param b
 *             Bicycle objektet att ta bort
 * @return true om det lyckades, annars false
 */
public boolean removeBicycle(Bicycle b)

/**
 * Tar bort ett valt User objekt från disken.
 *
 * @param u
 *             User objektet att ta bort
 * @return true om det lyckades, annars false
 */
public boolean removeUser(User u)

/**
 * Läser in alla Bicycle objekt som finns sparade på disken och
     returnerar
 * dessa.
 *
 * @return En vektor med alla Bicycle objekt som gick att läsa från
     disken
 */
public Bicycle[] loadBicycles()

/**
 * Läser in alla User objekt som finns sparade på disken och returnerar
 * dessa.
 *
 * @return En vektor med alla User objekt som gick att läsa från disken
 */
public User[] loadUsers()
```

3.4 DatabaseManager

3.4.1 Beskrivning

Ansvarig för klassen: Mattias Nordahl

Detta är klassen som hanterar alla operationer på i första hand grupper av Bicycle eller User objekt, t ex sortering av eller sökning bland dessa. Dess huvuduppgift är att skicka vidare och ta emot datan som BicycleGarageManager klassen begär eller skickar.

Varje instans av klassen är kopplad till en specifik instans av FileManager- och BicycleGarageManager-klasserna. Den ska se till att det inte blir onödigt mycket skrivande eller läsande från disk och därför hålla data som redan har hämtats i sin interna struktur. Denna hantering sker dolt utifrån, metodanrop

som kommer utifrån ska inte behöva bry sig om ifall datan ska hämtas med hjälp av FileManager klassen eller om den redan finns sparad internt. Innehåller även viss loggning av operationerna som utförs.

3.4.2 Konstruktorer

```
/**
 * Skapar en DatabaseManager bunden till en specifik FileManager.
 *
 * @param fm
 *              FileManager som denna DatabaseManager kommer att jobba mot
 * @param log
 *              Logger objekt som kommer att användas av klassen vid
     loggning
 */
public DatabaseManager(FileManager fm, Logger log)
```

3.4.3 Metoder

```
/**
 * Hämtar en cykel från databasen med en specifik BarCode.
 *
 * @param i
 *              BarCode på det Bicycle objekt som söks
 * @return Bicycle objektet med BarCode i om ett sådant finns, annars
     null
 */
public Bicycle getBicycleFromBarCode(int i)

/**
 * Hämtar alla cyklar från databasen som ägs av en viss användare.
 *
 * @param u
 *              Det User objekt vars Bicycle objekt söks
 * @return en vektor med alla Bicycle objekt som hittades
 */
public Bicycle[] getBicycleFromOwner(User u)

/**
 * Hämtar en användare från databasen med ett specifikt personnummer.
 *
 * @param i
 *              Personnumret på det User objekt som söks.
 * @return User objektet med personnummer i om ett sådant finns, annars
     null
 */
public User getUserFromPersonalNumber(long i)

/**
 * Hämtar alla användare från databasen med ett specifikt namn.
 *
 * @param s
 *              Namnet på den användare som söks
 * @return en vektor med alla User objekt som hittades
 */
public User[] getUserFromName(String s)


/**
 * Hämtar en användare från databasen med valda 6 första siffror i
 * personnummret samt en viss PIN-kod.
 *
 * @param personalNumber
 *              De 6 första siffrorna i personnumret på det User objekt
     som
 *              söks
 * @param pin
 *              Pin-koden för det User objekt som söks
 * @return User objektet med som matchar villkoren ovan om ett sådant
     finns,
 *          annars null
```

```
 */
public User getUserFromPersonalNumberAndPinCombination(long
    personalNumber,      int pin)

/**
 * Försöker hitta alla cyklar som har liknelser med strängen som
     skickades
 * in. Sökningar görs i attributen: barCode, frameNumber, owner,
     location,
 * description
 *
 * @param s
 *             Strängen som det ska sökas efter
 * @return en vektor med alla Bicycle objekt som liknar strängen s
 */
public Bicycle[] searchBicycle(String s)

/**
 * Försöker hitta alla användare som har liknelser med strängen som
 * skickades in. Sökningar görs i attributen: personalNumber, name,
 * nbrOfBicycles, lastVisit, nbrOfVisits
 *
 * @param s
 *             Strängen som det ska sökas efter
 * @return en vektor med alla User objekt som liknar strängen s
 */
public User[] searchUser(String s)

/**
 * Hämtar alla användare från databasen.
 *
 * @return En vektor med alla User objekt i databasen
 */
public User[] getAllUsers()
/**
 * Hämtar alla cyklar från databasen.
 *
 * @return En vektor med alla Bicycle objekt i databasen
 */
public Bicycle[] getAllBicycles()

/**
 * Lägger till en användare i databasen.
 *
 * @param u
 *             Det User objekt som vill läggas till
 * @return true om tillägget lyckades, annars false
 */
public boolean addUser(User u)

/**
 * Lägger till en cykel i databasen.
 *
 * @param b
 *             Det Bicycle objekt som vill läggas till
 * @return true om tillägget lyckades, annars false
 */
public boolean addBicycle(Bicycle b)

/**
 * Tar bort vald användare från databasen.
 *
 * @param u
 *             Det User objekt som önskas tas bort
 * @return true om borttagningen lyckades, annars false
 */
public boolean removeUser(User u)

/**
 * Tar bort vald cykel från databasen.
 *
 * @param u
 *             Det Bicycle objekt som önskas tas bort
 * @return true om borttagningen lyckades, annars false
```

```
 */
public boolean removeBicycle(Bicycle b)

/**
 * Ser till att vald cykel sparas i FileManagern som är bunden till
    denna
 * DatabaseManager. Bra att använda efter att ändringar har gjorts för
    att
 * försäkra sig om att de är sparade på disken.
 *
 * @param b
 *             Det Bicycle objekt som önksas sparas
 * @return true om sparningen lyckades, annars false
 */
public boolean saveBicycle(Bicycle b)

/**
 * Ser till att vald användare sparas i FileManagern som är bunden till
 * denna DatabaseManager. Bra att använda efter att ändringar har gjorts
     för
 * att försäkra sig om att de är sparade på disken.
 *
 * @param u
 *             Det User objekt som önksas sparas
 * @return true om sparningen lyckades, annars false
 */
public boolean saveUser(User u)
```

3.5 BicycleGarageManager

3.5.1 Beskrivning

Ansvarig för klassen: Mattias Nordahl

Detta är klassen som är hjärtat i programmet. Den binder ihop hårdvaruinterfacen med resten av systemet. Klassen innehåller metoder som hårdvarudrivrutiner anropar när de behöver skicka data till systemet. Den innehåller även metoder som BicycleGarageGUI anropar för att utföra olika kommandon, då väldigt lite arbete utförs direkt i klassen BicycleGarageGUI. Viss del av arbetet skickas vidare ytterliggare ett steg till klassen DatabaseManager. Klassen implementerar interfacet Obersvable.

Till varje instans av klassen är det bundet ett DatabaseManager objekt samt en fullständig uppsättning hårdvara. Hanterar en stor del av loggningen.

Dessutom innehåller klassen konstanter som används vid sortering och jämförelse av Bicycle och User objekt av andra klasser i systemet.

3.5.2 Konstruktorer

```
/**
 * Skapar en BicycleGarageManager som är bunden till en viss
 * DatabaseManager.
 *
 * @param dbm
 *             DatabaseManager som denna BicycleGarageManager kommer att
 *             jobba mot
 * @param log
 *             Logger objekt som kommer att användas av klassen vid
    loggning
 */
public BicycleGarageManager(DatabaseManager dbm, Logger log)
```

3.5.3 Metoder

```
/**
 * Register hardware so that BicycleGarageManager knows which drivers to
```

```
     * access.
     */
    public void registerHardwareDrivers(BarcodePrinter printer,
        ElectronicLock enrtyLock, PinCodeTerminal terminal)

    /**
     * Will be called when a user has the bar code reader at the entry door.
     * Bicycle ID should be a string of 5 characters, where every character
        can
     * be '0','1',... '9'.
     */
    public void entryBarcode(String bicycleID)

    /**
     * Will be called when a user has used the bar code reader at the exit
        door.
     * Bicycle ID should be a string of 5 characters, where every character
        can
     * be '0','1',... '9'.
     */
    public void exitBarcode(String bicycleID)

    /**
     * Will be called when a user has used the bar code reader at the exit
        door.
     * Bicycle ID should be a string of 5 characters, where every character
        can
     * be '0','1',... '9', '*', '#'.
     */
    public void entryCharacter(char c)

    /**
     * Anropas från GUI:t. Skapar en ny användare och lägger till den i
     * databasen.
     *
     * @param personalNumber
     *              Personnumret på användaren
     * @param name
     *              För- och efternamn på användaren
     * @param homePhone
     *              Telefon hem till användaren
     * @param mobilePhone
     *              Mobiltelefon för användaren
     * @param address
     *              Adress till användaren
     * @param notes
     *              Anteckningar om användaren
     * @return User objektet som skapades, null om det misslyckades
     */
    public User newUser(long personalNumber, String name, String homePhone,
        String mobilePhone, String address, String notes)

    /**
     * Anropas från GUI:t. Skapar en ny cykel och lägger till den i
        databasen.
     *
     * @param description
     *              Beskrivning av cykeln
     * @param frameNumber
     *              Eventuellt ramnummer på cykeln
     * @param owner
     *              Ägare av cykeln
     * @param notes
     *              Anteckningar om cykeln
     * @return Bicycle objektet som skapades, null om det misslyckades
     */
    public Bicycle newBicycle(String description, String frameNumber, User
        owner, String notes)

    /**
     * Anropas från GUI:t. Uppdaterar datan som finns sparad om en cykel.
     *
     * @param b
     *              Bicycle-objektet som ska uppdateras
```

```
     * @param description
     *              Beskrivning av cykeln
     * @param frameNumber
     *              Eventuellt ramnummer på cykeln
     * @param location
     *              Cyeklns status, 1 för i garaget, 0 för inte i garaget
     * @param notes
     *              Anteckningar om cykeln
     * @return true om det lyckades, annars false
     */
    public boolean editBicycle(Bicycle b, String description, String
        frameNumber, int location, String notes)

    /**
     * Anropas från GUI:t. Uppdaterar datan som finns sparad om en användare
         .
     *
     * @param u
     *              User objektet som ska uppdateras
     * @param personalNumber
     *              Personnumret på användaren
     * @param name
     *              För- och efternamn på användaren
     * @param homePhone
     *              Telefon hem till användaren
     * @param mobilePhone
     *              Mobiltelefon för användaren
     * @param address
     *              Adress till användaren
     * @param notes
     *              Anteckningar om användaren
     * @return true om det lyckades, annars false
     */
    public boolean editUser(User u, int personal_number, String name, String
         home_phone, String mobile_phone, String address, String notes)

    /**
     * Anropas från GUI:t. Försöker generera en ny PIN-kod, till en
         användare.
     * PIN-koden kommer garanterat att vara unik för varje användargrupp med
     * samma 6 första siffror i personnumret.
     *
     * @param u
     *              User-objektet som ska få en ny PIN-kod
     * @return true om det lyckades, annars false
     */
    public boolean assignNewPinCode(User u)

    /**
     * Anropas från GUI:t. Sparar en användare till databasen.
     *
     * @param u
     *              User-objektet som ska sparas
     * @return true om det lyckades, annars false
     */
    public boolean save(User u)

    /**
     * Anropas från GUI:t. Sparar en cyckel till databasen
     *
     * @param u
     *              Bicycle-objektet som ska sparas
     * @return true om det lyckades, annars false
     */
    public boolean save(Bicycle b)

    /**
     * Anropas från GUI:t. Tar bort vald cykel från systemet.
     *
     * @param b
     *              Bicycle-objektet som önskas tas bort
     * @return true om det lyckades, annars false
     */
    public boolean removeBicycle(Bicycle b)
```

```
/**
 * Anropas från GUI:t. Tar bort vald användare från systemet.
 *
 * @param u
 *             User-objektet som önskas tas bort
 * @return true om det lyckades, annars false
 */
public boolean removeUser(User u)

/**
 * Sorterar alla användare enligt vald ordning och returnerar dessa.
    Använd
 * konstanterna i denna klass.
 *
 * @param u
 *             En vektor av User-objekt att sortera
 * @param sortOrder
 *             Ordningen att sortera efter. Använd en av konstanterna i
 *             denna klass.
 * @return En vektor av User-objekt sorterad efter ordningen sortOrder
 */
private User[] sortUsers(User[] u, int sortOrder)

/**
 * Sorterar alla cyklar enligt vald ordning och returnerar dessa. Använd
 * konstanterna i denna klass.
 *
 * @param u
 *             En vektor av Bicycle-objekt att sortera
 * @param sortOrder
 *             Ordningen att sortera efter. Använd en av konstanterna i
 *             denna klass.
 * @return En vektor av Bicycle-objekt sorterad efter ordningen
    sortOrder
 */
private Bicycle[] sortBicycles(Bicycle[] b, int sortOrder)

/**
 * Hämtar alla användare som finns i systemet i en viss ordning. Använd
 * konstanterna i denna klass.
 *
 * @param sortOrder
 *             Ordningen att sortera efter. Använd en av konstanterna i
 *             denna klass.
 * @return En vektor av User-objekt sorterad efter ordningen sortOrder
 */
public User[] getAllUsers(int sortOrder)

/**
 * Hämtar alla cyklar som finns i systemet i en viss ordning. Använd
 * konstanterna i denna klass.
 *
 * @param sortOrder
 *             Ordningen att sortera efter. Använd en av konstanterna i
 *             denna klass.
 * @return En vektor av Bicycle-objekt sorterad efter ordningen
    sortOrder
 */
public Bicycle[] getAllUser(int sortOrder)

/**
 * Gör en sökning efter användare genom att titta i fälten
    personalNumber,
 * name, nbrOfBicycles, lastVisit, nbrOfVisits. Sökresultaten kommer
 * tillbaka i vald ordning. Använd konstanterna i denna klass.
 *
 * @param s
 *             Strängen som det ska sökas efter
 * @param sortOrder
 *             Ordningen att sortera efter. Använd en av konstanterna i
 *             denna klass.
 * @return En vektor av User-objekt sorterad efter ordningen sortOrder
 */
```

```
public User[] searchUser(String s, int sortOrder)

/**
 * Gör en sökning efter cyklar genom att titta i fälten barCode,
 * frameNumber, owner, location, description. Sökresultaten kommer
     tillbaka
 * i vald ordning. Använd konstanterna i denna klass.
 *
 * @param s
 *              Strängen som det ska sökas efter
 * @param sortOrder
 *              Ordningen att sortera efter. Använd en av konstanterna i
 *              denna klass.
 * @return En vektor av Bicycle-objekt sorterad efter ordningen
     sortOrder
 */
public Bicycle[] searchBicycle(String s, int sortOrder)

/**
 * Skickar ett jobb till skrivaren som innehåller en begäran om en
     utskrift
 * av streckkoden till den aktuella cykeln.
 *
 * @param b
 *              Det Bicycle-objekt som ska få en ny streckkod utskriven
 * @return true om det lyckades, annars false
 */
public boolean printBarCode(Bicycle b)

/**
 * Hämtar användbar statistik om systemet.
 *
 * @return en Map<String, String> där den första strängen innehåller
 *          information om statistiken och den andra strängen anger
     värdet.
 *          En viss information återfinns alltid på samma plats.
 */
public Map<String, String> getStatistics()

/**
 * Anropas från GUI:t. Flyttar en cyckel in eller ut ur garaget.
 *
 * @param b
 *              Bicycle objektet som ska flyttas
 * @param location
 *              Cyckelns status, 1 för i garaget eller 0 för ute ur
     garaget
 * @return true om det lyckades, annars false
 */
public boolean setBicycleLocation(Bicycle b, int location)
```

3.6 BicycleGarageGUI

3.6.1 Beskrivning

Ansvarig för klassen: Hannes Nevalainen

Detta är klassen som hanterar det grafiska användargränssnittet. Klassen utför inga operationer själv utan skickar alla dessa vidare till den BicycleGarageManager som den blev bunden till vid skapandet. Notera att det inte finns någon funktion för loggning i denna klass då allt arbete utförs av det BicycleGarageManager-objekt som den är bunden till. Klassen implementerar interfacet Observer.

3.6.2 Konstruktorer

```
/**
 * Skapar ett BicycleGarageGUI som är bundet till en viss
 * BicycleGarageManager.
```

```
     *
     * @param bgm
     *              BicycleGarageManager som denna BicycleGarageGUI kommer att
     *              jobba mot
     */
    public BicycleGarageGUI(BicycleGarageManager bgm)

    /**
     * This method is called whenever the observed object is changed. An
     * application calls an Observable object's notifyObservers method to
         have
     * all the object's observers notified of the change.
     *
     * @param obs
     *              the observable object.
     * @param type
     *              an argument passed to the notifyObservers method.
     *
     */
    public void update(Observable obs, Object type)
```

3.6.3 Metoder

```
    @override
    public void update(Observable arg0, Object arg1)
```

3.7 CompareBicycle

3.7.1 Beskrivning

Ansvarig för klassen: Daniel Olofsson och Sandra Nilsson

Denna klass har som syfte att hjälpa till vid sorteringen av objekt av klassen Bicycle. Detta gör den genom att fungera som en representation av interfacet Comparator för den här typen av objekt. Den använder sig av de konstanter som finns fördefinerade i klassen BicycleGarageManager.

3.7.2 Konstruktorer

```
    /**
     * Skapar en komparator som är avsedd att jämföra Bicycle objekt.
     */
    public CompareBicycle()
```

3.7.3 Metoder

```
    /**
     * Ändrar vilket fält som jämförelse sker efter. Använd konstanterna i
     * BicycleGarageManager klassen.
     *
     * @param i
     *              Fält att jämföra. Använd en av konstanterna i
     *              BicycleGarageManager klassen
     */
    public void setCompare(int i)

    @Override
    public int compare(Bicycle o1, Bicycle o2)
```

3.8 CompareUser

3.8.1 Beskrivning

Ansvarig för klassen: Emil Einarsson

Denna klass har som syfte att hjälpa till vid sorteringen av objekt av klassen User. Detta gör den genom att fungera som en representation av interfacet

Comparator för den här typen av objekt. Den använder sig av de konstanter som finns fördefinerade i klassen BicycleGarageManager.

3.8.2 Konstruktorer

```
/**
 * Skapar en komparator som är avsedd att jämföra User objekt.
 */
public CompareUser()
```

3.8.3 Metoder

```
/**
 * Ändrar vilket fält som jämförelse sker efter. Använd konstanterna i
 * BicycleGarageManager klassen.
 *
 * @param i
 *             Fält att jämföra. Använd en av konstanterna i
 *             BicycleGarageManager klassen
 */
public void setCompare(int i)

@Override
public int compare(User o1, User o2)
```

3.9 BicycleGarage

3.9.1 Beskrivning

Ansvarig för klassen: Fredrik Andersson

Innehåller main-metoden vilken knyter ihop säcken. Skapar objekt av alla nödvändiga klasser (inklusive TestDrivers) och skickar med dessa som parametrar där det behövs. Slutligen startas programmet och GUI:t visas.

3.9.2 Konstruktorer

```
public BicycleGarage()
```

3.9.3 Metoder

```
/**
 * @param args
 */
public static void main(String[] args)
```