Tampereen ammattikorkeakoulu

Ohjelmoinnin syventävät tekniikat

Sakari Liesjärvi

SISÄLLYS

1 Viikko

1.1 Tehtävä 1

```
public class randomNumber {
    public static void main(String[] args) {
        int x = (int) Math.floor(Math.random() * (100 - 1) + 1);
        System.out.printf("Luku, jota ajattelen on %d%n", x);
    }
}
```

```
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java$ java randomNumber
Luku, jota ajattelen on 56
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java$ java randomNumber
Luku, jota ajattelen on 12
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java$
```

Ohjelma arpoo int -tyyppisen kokonaisluvun väliltä 1–100 ja tulostaa tämän.

1.2 Tehtävä 2

```
import java.util.Scanner;

public class guessNumber {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int counter = 1;
        boolean correct=false;

        int random = (int) Math.floor(Math.random() * (100 - 1) + 1);

        do {
            System.out.printf("%nGuess the number between 1 and 100: ");
            int guess = scanner.nextInt();

            if (guess == random) {
                System.err.printf("%nCongratulations, the correct number
was %d%n", random);
                correct=true;
            }
            else if (counter < 7) {
                System.out.printf("%nWrong! Guesses left: %d%n", 7-coun-
ter);
                counter++;
            }
            else {
                System.out.printf("%nYou ran out of guesses, the correct
number was %d%n", random);
                correct=true;
```

```
            }
        } while(!correct);

        scanner.close();
    }
}
```

```
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java$ javac guessNumber.java
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java$ java guessNumber

Guess the number between 1 and 100: 7

Wrong! Guesses left: 6

Guess the number between 1 and 100: 10

Wrong! Guesses left: 5

Guess the number between 1 and 100: 39

Wrong! Guesses left: 4

Guess the number between 1 and 100: 86

Wrong! Guesses left: 3

Guess the number between 1 and 100: 98

Wrong! Guesses left: 2

Guess the number between 1 and 100: 56

Wrong! Guesses left: 1

Guess the number between 1 and 100: 27

You ran out of guesses, the correct number was 4
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java$
```

```
Guess the number between 1 and 100: 20

Wrong! Guesses left: 4

Guess the number between 1 and 100: 98

Congratulations, the correct number was 98
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java$
```

Ohjelma antaa käyttäjälle 7 yritystä arvata satunnainen luku väliltä 1–100. Tämän lisäksi ohjelma näyttää jäljellä olevien arvausten määrän.

2 Viikko

2.1 Tehtävä 3

TimeOnly.java:

```
public class TimeOnly {
    private int hours;
    private int minutes;
    private int seconds;

    public TimeOnly(int hours, int minutes, int seconds) {
        this.hours = hours;
        this.minutes = minutes;
        this.seconds = seconds;
    }

    public int getHours() {
        return hours;
    }

    public int getMinutes() {
        return minutes;
    }

    public int getSeconds() {
        return seconds;
    }

    public void setHours(int hours) {
        if(hours < 0 || hours > 23) {
            throw new IllegalArgumentException("Hours must be between
0...23");
        }
        this.hours = hours;
    }

    public void setMinutes(int minutes) {
        if (minutes < 0 || minutes > 59) {
            throw new IllegalArgumentException("Minutes must be between
0...59");
        }
        this.minutes = minutes;
    }

    public void setSeconds(int seconds) {
        if (seconds < 0 || seconds > 59) {
            throw new IllegalArgumentException("Seconds must be between
0...59");
```

```
        }
        this.seconds = seconds;
    }

    @Override
    public String toString() {
        return String.format("%02d:%02d:%02d", hours, minutes, seconds);
    }
}
```

TimeOnlyTest.java:

```
public class TimeOnlyTest {
    public static void main(String[] args) {
        TimeOnly t = new TimeOnly(0, 0, 0);
        TimeOnly t2 = new TimeOnly(0, 0, 0);

        try {
            t.setMinutes(44);
            t.setSeconds(38);
            t.setHours(13);
        }
        catch (IllegalArgumentException iae) {
            System.err.println("Problems with TimeOnly: " + iae.getMes-
sage());
        }
        System.out.println(t);

        try {
            t2.setMinutes(60);
            t2.setSeconds(59);
            t2.setHours(24);
        }
        catch (IllegalArgumentException iae) {
            System.err.println("Problems with TimeOnly: " + iae.getMes-
sage());
        }
        System.out.println(t2);
    }
}
```

Ohjelmassa on yksinkertainen tarkistus settereissä, jotta tunnit ovat aina välillä 0-23 ja minuutit sekä sekunnit 0-59.

```
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java/viikko2$ java TimeOnlyTest.java
13:44:38
Problems with TimeOnly: Minutes must be between 0...59
00:00:00
```

Ajosta nähdään, että ensimmäiselle oliolle pystytään asettamaan järkevä aika settereillä mutta toiselle tulee virhe, koska minuutteihin oli yritetty asettaa 60.

```
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java/viikko2$ java TimeOnlyTest.java
13:44:38
Problems with TimeOnly: Hours must be between 0...23
00:25:17
```

Nyt tunteihin yritettiin asettaa 24.

2.2 Tehtävä 4

Timestamp.java:

```java
import java.util.StringTokenizer;

public class Timestamp {
    private DateOnly date;
    private TimeOnly time;

    public Timestamp(DateOnly date, TimeOnly time) {
        this.date = date;
        this.time = time;
    }

    public DateOnly getDate() {
        return date;
    }

    public TimeOnly getTime() {
        return time;
    }

    @Override
    public String toString() {

        // Haetaan päivämäärä ja aika luokkien gettereillä
        DateOnly date = getDate();
        TimeOnly time = getTime();

        // Formatoidaan päivämäärä ja aika luokkien metodeilla
        String formattedDate = date.toString();
        String formattedTime = time.toString();

        // Palautetaan timestamp oikeassa formaatissa
        return formattedDate + " " + formattedTime;
    }

    public static Timestamp parse(String s) {
```

```
        // Luodaan uusi StringTokenizer, joka erottaa stringistä kaksi
osaa T kirjaimella
        StringTokenizer tokenizer = new StringTokenizer(s, "T");

        // Jos stringistä ei löydy kahta osaa (date ja time)
        if (tokenizer.countTokens() != 2) {
            throw new IllegalArgumentException("Invalid timestamp for-
mat");
        }

        // Erotetaan date ja time
        String dateString = tokenizer.nextToken();
        String timeString = tokenizer.nextToken();


        try {
            // Jaetaan dateString ja timeString kokonaislukuosiin, joita
voidaan käyttää luokan konstruktorissa
            String[] dateParts = dateString.split("-");
            int year = Integer.parseInt(dateParts[0]);
            int month = Integer.parseInt(dateParts[1]);
            int day = Integer.parseInt(dateParts[2]);

            String[] timeParts = timeString.split(":");
            int hour = Integer.parseInt(timeParts[0]);
            int minute = Integer.parseInt(timeParts[1]);
            int second = Integer.parseInt(timeParts[2]);

            // Luodaan uudet oliot saaduilla arvoilla. Asetetaan arvot
myös settereillä,
            // joista löytyy virheentarkistus.
            DateOnly date = new DateOnly(year, month, day);
            date.setYear(year);
            date.setMonth(month);
            date.setDay(day);

            TimeOnly time = new TimeOnly(hour, minute, second);
            time.setHours(hour);
            time.setMinutes(minute);
            time.setSeconds(second);

            // Palautetaan valmis timestamp
            return new Timestamp(date, time);
        }
        catch (IllegalArgumentException iae) {
            throw new IllegalArgumentException(iae.getMessage());
        }

    }
}
```

TimestampTest.java:

```
public class TimestampTest {
    public static void main(String[] args) {
        // Luodaan yksinkertainen timestamp
        DateOnly d = new DateOnly(2024, 1, 24);
        TimeOnly t = new TimeOnly(14, 2, 38);
        Timestamp ts = new Timestamp(d, t);

        System.out.println(ts);

        // Kokeillaan luoda ts2 timestamp stringistä
        try {
            Timestamp ts2 = Timestamp.parse("2024-01-08T09:45:00");
            System.out.println(ts2);
        }
        catch (IllegalArgumentException iae) {
            System.err.println("Problems with timestamp: " + iae.getMes-
sage());
        }

        // Luodaan virheellinen timestamp testin vuoksi
        try {
            Timestamp ts3 = Timestamp.parse("2024-01-08 09:45:00");
            System.out.println(ts3);
        }
        catch (IllegalArgumentException iae) {
            System.err.println("Problems with timestamp: " + iae.getMes-
sage());
        }
    }
}
```

```
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java/viikko2$ java TimestampTest.java
2024-01-24 14:02:38
2024-01-08 09:45:00
Problems with timestamp: Invalid timestamp format
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java/viikko2$
```

```
Timestamp ts3 = Timestamp.parse(s:"2024-03-13 13:28:00");
```

Ajosta nähdään, että timestamp olio voidaan luoda konstruktorilla onnistuneesti ja myös parsella stringistä. Nähdään myös virheilmoitus, kun stringi onkin väärässä muodossa.

```
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java/viikko2$ java TimestampTest.java
2024-01-24 14:02:38
2024-01-08 09:45:00
Problems with timestamp: Day must be between 1...31 depending on month
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java/viikko2$
```

```
Timestamp ts3 = Timestamp.parse(s:"2024-02-30T40:28:00");
```

```
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java/viikko2$ java TimestampTest.java
2024-01-24 14:02:38
2024-01-08 09:45:00
Problems with timestamp: Hours must be between 0...23
```

```
Timestamp ts3 = Timestamp.parse(s:"2024-02-28T40:28:00");
```

Nähdään myös, että parse metodi ottaa huomioon DateOnly ja TimeOnly luokkien setterien rajoitukset.

3 Viikko

3.1 Tehtävä 5

Java.time pakkaus antaa kasan valmiita funktioita, jotka ovat käteviä ja valmiita ratkaisuja. Myös virheellisten päivämäärien / aikojen syöttö on estetty valmiiksi. Käyttö voi olla nopeampaa ja joustavampaa monissa tilanteissa kuin itse tehdyt luokat. Java.time tekee monimutkaisemmissa projekteissa koodista myös usein helpompaa luettavaa.

Kuitenkin omien luokkien käyttäminen antaa enemmän muotoiltavaa ja täyden hallinnan. Käyttö voi olla helpompaa yksinkertaisissa projekteissa, joissa ei tarvita edistyneempiä toimintoja.

3.2 Tehtävä 6

Test.java:

```
import java.util.Arrays;
import java.util.Collections;
import java.util.List;

public class Test {
    public static void main(String[] args) {
        // Luodaan Event oliot
        Event event1 = new Event("2024-01-31", "kuvaus", "kategoria1");
        Event event2 = new Event("2023-12-24", "kuvaus2", "kategoria2");
        Event event3 = new Event("2024-01-29", "kuvaus3", "kategoria3");

        // Luodaan eventeistä taulukkolista
        List<Event> events = Arrays.asList(event1, event2, event3);
        // Järjestellään lista compareTo metodia käyttäen
        Collections.sort(events);

        // Tulostetaan kaikki eventit
        for (Event event : events) {
            System.out.println(event + "\n");
        }
    }
}
```

Event.java:

```
import java.time.LocalDate;
```

```
public class Event implements Comparable<Event> {
    private LocalDate date;
    private Description description;
    private Category category;

    public Event(String date, String description, String category) {
        // Käytetään LocalDaten parsea stringiin
        setDate(LocalDate.parse(date));

        // Luodaan uusi kuvaus ja kategoria
        // Tällöin käytetään luokkien omia settereitä ja virheentarkis-
tusta
        this.description = new Description(description);
        this.category = new Category(category);
    }

    public LocalDate getDate() {
        return this.date;
    }

    public void setDate(LocalDate date) {
        this.date = date;
    }

    // Muotoillaan tulostus
    @Override
    public String toString() {
        return "Event: " +
                "\n date: " + date +
                "\n description: " + description +
                "\n category: " + category;
    }

    @Override
    public int compareTo(Event other) {
        // Ensin vertaillaan päivämääriä
        if (this.date.compareTo(other.date) == 0) {
            // Vertaillaan kuvauksia, jos päivämäärät olivat samat
            if (this.description.compareTo(other.description) == 0) {
                // Palautetaan luokituksien vertailun arvo
                return this.category.compareTo(other.category);
            }
            // Palautetaan kuvauksen vertailun arvo jos kuvaukset eivät
samat
            return this.description.compareTo(other.description);
        }
        // Jos päivämäärät eivät ole samoja, palautetaan niiden vertailun
arvo
        return this.date.compareTo(other.date);
    }
```

```
}
```

Description.java:

```
public class Description implements Comparable<Description> {
    private String desc;

    public Description(String desc) {
        setDescription(desc);
    }

    public String getDescription() {
        return this.desc;
    }

    public void setDescription(String desc) {
        // Tarkistetaan, että kuvaus ei ole null, tyhjä ja max 500 merk-
kiä pitkä
        if (desc != null && !desc.isEmpty() && desc.length() <= 500) {
            this.desc = desc;
        }
        else {
            throw new IllegalArgumentException("Invalid description");
        }
    }

    // compareTo metodi lajittelua varten
    @Override
    public int compareTo(Description other) {
        return this.desc.compareTo(other.desc);
    }

    @Override
    public String toString() {
        return desc;
    }
}
```

Category.java:

```
public class Category implements Comparable<Category>{
    private String category;

    public Category(String category) {
        setGategory(category);
    }

    public String getGategory() {
        return this.category;
    }
```

```
    public void setGategory(String category) {
        // Tarkistetaan, että kategoria ei voi olla null, tyhjä ja on max
50 merkkiä
        if (category != null && !category.isEmpty() && category.length()
<= 50) {
            // Tarkistetaan, että kategoria on kirjoitettu pienellä
            if (category == category.toLowerCase()) {
                this.category = category;
            }
            else {
                throw new IllegalArgumentException("Category must be in
lower case");
            }
        }
        else {
            throw new IllegalArgumentException("Invalid category");
        }
    }

    // compareTo metodi lajittelua varten
    @Override
    public int compareTo(Category other) {
        return this.category.compareTo(other.category);
    }

    @Override
    public String toString() {
        return category;
    }
}
```

Ajo esimerkki:

```
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java/viikko3$ javac Category.java Description.java Event.java Test.java
zage@LAPTOP-A82R9UN0:~/OhSyTe/Java/viikko3$ java Test
Event:
 date: 2023-12-24
 description: kuvaus2
 category: kategoria2

Event:
 date: 2024-01-29
 description: kuvaus3
 category: kategoria3

Event:
 date: 2024-01-31
 description: kuvaus
 category: kategoria1
```

Ohjelma tulostaa kaikki luodut tapahtumat ja niiden tiedot. Ennen tulostusta hyödynnetään olioiden vertailua ja järjestetään tapahtumat päivämäärän mukaan.

Jos päivämäärät ovat samat, järjestetään luokat kuvauksen mukaan ja viimeisessä tapauksessa kategorian mukaan.

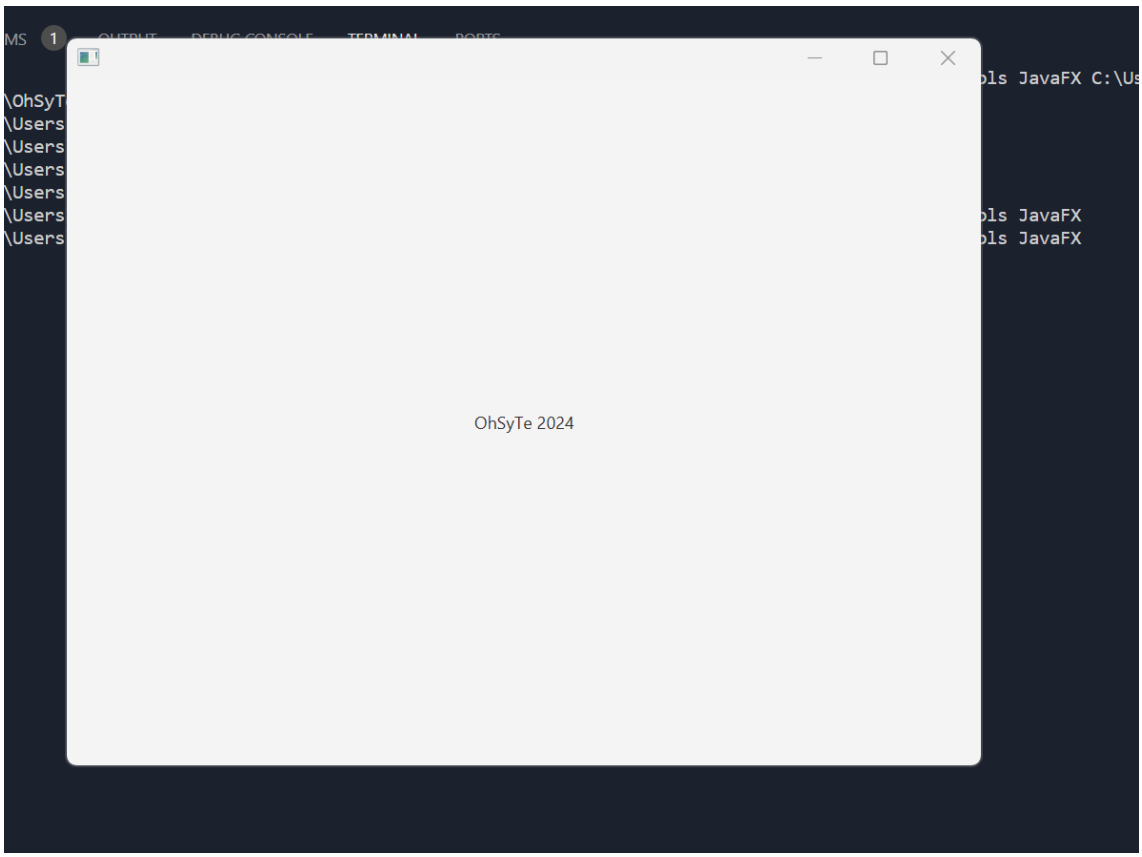
4 Viikko

4.1 Tehtävä 7

```java
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.layout.StackPane;
import javafx.stage.Stage;
// ^ Nämä importit tarvitaan

// JavaFX-luokka laajentaa Application-luokkaa ja mahdollistaa sovellus-
ten luomisen
public class JavaFX extends Application {

    @Override
    public void start(Stage stage) {
        // Luodaan teksti
        Label l = new Label("OhSyTe 2024");

        // Luodaan asetelma, joka laitetaan stage-ikkunaan
        Scene scene = new Scene(new StackPane(l), 640, 480);
        stage.setScene(scene);
        stage.show();
    }

    public static void main(String[] args) {
        launch();
    }

}
```

Ohjelman ajo:

```
PS C:\Users\saket\tamk\OhSyTe\viikko4>
PS C:\Users\saket\tamk\OhSyTe\viikko4> java --module-path C:/javafx/lib --add-modules javafx.controls JavaFX
```



Ohjelmaa ajettaessa aukeaa ikkuna, jossa lukee ”OhSyTe 2024”

4.2 Tehtävä 8

ShowEvents.java:

```
import java.time.LocalDate;
import java.util.Collections;
import javafx.application.Application;
import javafx.collections.FXCollections;
import javafx.scene.Scene;
import javafx.scene.control.Label;
import javafx.scene.control.ListView;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class ShowEvents extends Application {
    private EventManager eventManager = EventManager.getInstance();

    @Override
    public void start(Stage stage) {
        addEvents(eventManager);
```

```
        // Lajitellaan lista päivämäärän mukaan
        Collections.sort(eventManager.getEvents(), Collections.reverseOr-
der());

        // Luodaan uusi listanäkymä
        ListView<Event> listView = new ListView<>(FXCollections.obser-
vableArrayList(eventManager.getEvents()));

        // Asetetaan listalle leveys ja korkeus
        listView.setPrefWidth(600);
        listView.setPrefHeight(400);

        // Lisätään sarakkeille otsikot
        Label dateLabel = new Label("Date");
        Label descriptionLabel = new Label("Description");
        Label categoryLabel = new Label("Category");

        // Luodaan HBox layout otsikoita varten
        HBox header = new HBox(dateLabel, descriptionLabel, catego-
ryLabel);
        header.setSpacing(100);

        // Luodaan VBox layout otsikoiden ja listanäkymän kanssa
        VBox root = new VBox(header, listView);

        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.show();
    }

    private void addEvents(EventManager em) {
        // Lisätään tapahtumat EventManageriin
        em.addEvent(
            new Event(LocalDate.parse("2020-11-12"),
            new Description("macOS 11 Big Sur released"),
            new Category("apple"))
            );
        em.addEvent(
            new Event(LocalDate.parse("2015-09-03"),
            new Description("OS X 10.11 El Capitan released"),
            new Category("apple"))
            );
        em.addEvent(
            new Event(LocalDate.parse("2019-10-07"),
            new Description("macOS 10.15 Catalina released"),
            new Category("apple"))
            );
        em.addEvent(
            new Event(LocalDate.parse("2017-09-25"),
            new Description("macOS 10.13 High Sierra released "),
            new Category("apple"))
```

```
        );
    em.addEvent(
        new Event(LocalDate.parse("2007-10-26"),
        new Description("Mac OS X 10.5 Leopard released"),
        new Category("apple"))
        );
    em.addEvent(
        new Event(LocalDate.parse("2011-07-20"),
        new Description("Mac OS X 10.7 Lion released"),
        new Category("apple"))
        );
    em.addEvent(
        new Event(LocalDate.parse("2013-10-22"),
        new Description("OS X 10.9 Mavericks released"),
        new Category("apple"))
        );
    em.addEvent(
        new Event(LocalDate.parse("2018-09-24"),
        new Description("macOS 10.14 Mojave released"),
        new Category("apple"))
        );
    em.addEvent(
        new Event(LocalDate.parse("2021-10-25"),
        new Description("macOS 12 Monterey released "),
        new Category("apple"))
        );
    em.addEvent(
        new Event(LocalDate.parse("2012-07-25"),
        new Description("OS X 10.8 Mountain Lion released"),
        new Category("apple"))
        );
    em.addEvent(
        new Event(LocalDate.parse("2016-09-20"),
        new Description("macOS 10.12 Sierra released"),
        new Category("apple"))
        );
    em.addEvent(
        new Event(LocalDate.parse("2009-08-28"),
        new Description("mMac OS X 10.6 Snow Leopard released"),
        new Category("apple"))
        );
    em.addEvent(
        new Event(LocalDate.parse("2023-09-26"),
        new Description("macOS 14 Sonoma released"),
        new Category("apple"))
        );
    em.addEvent(
        new Event(LocalDate.parse("2005-04-29"),
        new Description("Mac OS X 10.4 Tiger released"),
        new Category("apple"))
        );
```

```
        em.addEvent(
            new Event(LocalDate.parse("2022-10-24"),
            new Description("macOS 13 Ventura released "),
            new Category("apple"))
            );
        em.addEvent(
            new Event(LocalDate.parse("2014-10-16"),
            new Description("OS X 10.10 Yosemite released"),
            new Category("apple"))
            );
    }

    public static void main(String[] args) {
        // Käynnistetään sovellus
        launch(args);
    }
}
```

Event.java:

```
import java.time.LocalDate;

public class Event implements Comparable<Event> {
    private LocalDate date;
    private Description description;
    private Category category;

    public Event(LocalDate date, Description description, Category cate-
gory) {
        this.date = date;
        this.description = description;
        this.category = category;
    }

    public LocalDate getDate() {
        return this.date;
    }

    public Description getDescription() {
        return description;
    }

    public Category getCategory() {
        return category;
    }

    public void setDate(LocalDate date) {
        this.date = date;
    }
```

```
    // Muotoillaan tulostus
    @Override
    public String toString() {
        return String.format("%-12s %-40s %-10s", date, description, ca-
tegory);
    }

    @Override
    public int compareTo(Event other) {
        // Ensin vertaillaan päivämääriä
        if (this.date.compareTo(other.date) == 0) {
            // Vertaillaan kuvauksia, jos päivämäärät olivat samat
            if (this.description.compareTo(other.description) == 0) {
                // Palautetaan luokituksien vertailun arvo
                return this.category.compareTo(other.category);
            }
            // Palautetaan kuvauksen vertailun arvo jos kuvaukset eivät
samat
            return this.description.compareTo(other.description);
        }
        // Jos päivämäärät eivät ole samoja, palautetaan niiden vertailun
arvo
        return this.date.compareTo(other.date);
    }
}
```

EventManager.java:

```
import java.util.List;
import java.util.ArrayList;

//
// The "singleton" design pattern.
// Get the only instance by calling getInstance() in your program.
//

public class EventManager {
    private List<Event> events;

    // Private constructor, nobody else can create instances
    private EventManager() {
        this.events = new ArrayList<Event>();
    }

    // Private instance, statically created.
    private static final EventManager INSTANCE = new EventManager();

    // When someone wants a reference, they get it through this method.
    public static EventManager getInstance() {
```

```
        return INSTANCE;
    }

    // Any other methods can be public.

    public void addEvent(Event e) {
        this.events.add(e);
    }

    // It may not be a good idea to give a reference to the whole
    // event list. Maybe clone the list instead, and return the clone?
    public List<Event> getEvents() {
        return this.events;
    }
}
```

Description.java:

```
public class Description implements Comparable<Description> {
    private String desc;

    public Description(String desc) {
        setDescription(desc);
    }

    public String getDescription() {
        return this.desc;
    }

    public void setDescription(String desc) {
        // Tarkistetaan, että kuvaus ei ole null, tyhjä ja max 500 merk-
kiä pitkä
        if (desc != null && !desc.isEmpty() && desc.length() <= 500) {
            this.desc = desc;
        }
        else {
            throw new IllegalArgumentException("Invalid description");
        }
    }

    // compareTo metodi lajittelua varten
    @Override
    public int compareTo(Description other) {
        return this.desc.compareTo(other.desc);
    }

    @Override
    public String toString() {
        return desc;
    }
```

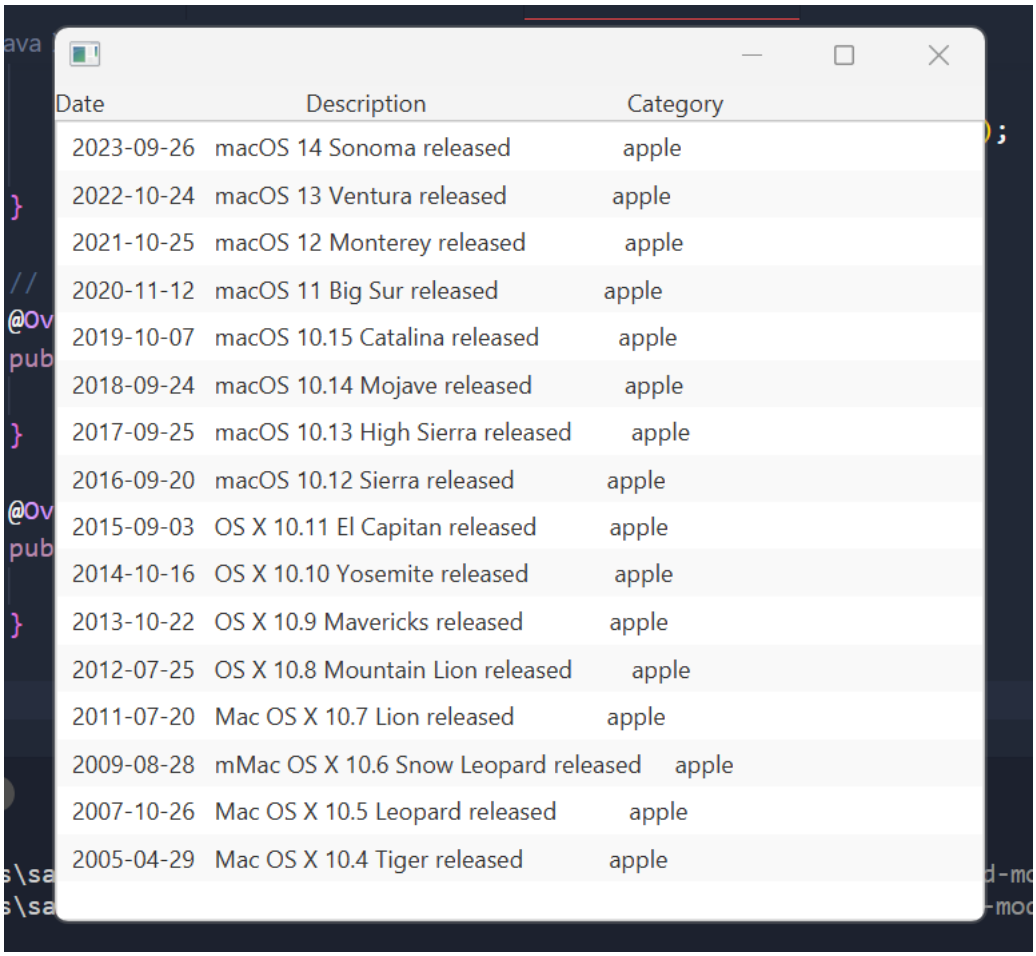
```
}
```

Category.java:

```
public class Category implements Comparable<Category>{
    private String category;

    public Category(String category) {
        setCategory(category);
    }

    public String getCategory() {
        return this.category;
    }

    public void setCategory(String category) {
        // Tarkistetaan, että kategoria ei voi olla null, tyhjä ja on max
50 merkkiä
        if (category != null && !category.isEmpty() && category.length()
<= 50) {
            // Tarkistetaan, että kategoria on kirjoitettu pienellä
            if (category == category.toLowerCase()) {
                this.category = category;
            }
            else {
                throw new IllegalArgumentException("Category must be in
lower case");
            }
        }
        else {
            throw new IllegalArgumentException("Invalid category");
        }
    }

    // compareTo metodi lajittelua varten
    @Override
    public int compareTo(Category other) {
        return this.category.compareTo(other.category);
    }

    @Override
    public String toString() {
        return category;
    }
}
```

Ohjelman ajo:

```
PS C:\Users\saket\tamk\OhSyTe\viikko4> javac --module-path C:/javafx/lib --add-modules javafx.controls ShowEvents.java
PS C:\Users\saket\tamk\OhSyTe\viikko4> java --module-path C:/javafx/lib --add-modules javafx.controls ShowEvents
```



5 Viikko

5.1 Tehtävä 9

Program.cs:

```
using System.Collections.Generic;
using System.Diagnostics.Tracing;

// Creating events
Event e1 = new Event(new DateOnly(2023,11,14), ".NET 8 released", "micro-
soft", "dotnet");
Event e2 = new Event(new DateOnly(2022,11,8), ".NET 7 released", "micro-
soft");
Event e3 = new Event(new DateOnly(2021,11,8), ".NET 6 released", "micro-
soft", "TEST");

// Creating events list
List<Event> events = new List<Event>();

// Adding events to the list
events.Add(e1);
events.Add(e2);
events.Add(e3);

// Printing before sorting
foreach (Event e in events) {
    Console.WriteLine(e);
}

events.Sort();

// Printing after sorting
Console.WriteLine("\nSorted events:");
foreach (Event e in events) {
    Console.WriteLine(e);
}
```

Event.cs:

```
public class Event: IComparable {
    // Automatic property: direct access to private field
    public DateOnly Date { get; set; }
    // Private backing field; validation in setter
    private string description;
    private Category EventCategory;

    public string Description {
```

```
        get {
            return this.description;
        }

        set {
            // The incoming new value is always named 'value'.
            ArgumentNullException.ThrowIfNullOrEmpty(value);

            if (value.Length > 500) {
                throw new ArgumentException("Description is too long!");
            }
            this.description = value;
        }
    }

    public Event(DateOnly date, string description, string primaryCate-
gory, string secondaryCategory = null) {
        this.Date = date;
        this.Description = description;
        this.EventCategory = new Category(primaryCategory, secondaryCate-
gory);
    }

    public override string ToString()
    {
        return $"{this.Date} {this.Description} ({this.EventCategory})";
    }

    // IComparable implementation
    public int CompareTo(object obj) {
        if (obj == null) {
            return 1;
        }

        Event otherEvent = obj as Event;
        if (otherEvent != null) {
            return this.Date.CompareTo(otherEvent.Date);
        }
        else {
            throw new ArgumentException("Object is not an event");
        }
    }
}
```

Category.cs:

```
public class Category {
    private string primary;
    private string secondary;

    public string Primary {
```

```
        get => this.primary;
        set {
            ArgumentNullException.ThrowIfNullOrEmpty(value);

            if (value.Length > 50) {
                throw new ArgumentException("Primary category is too
long!");
            }

            this.primary = value.ToLower();
        }
    }

    public string Secondary {
        get => this.secondary;
        set {
            ArgumentNullException.ThrowIfNullOrEmpty(value);

            if (value.Length > 50) {
                throw new ArgumentException("Secondary category is too
long!");
            }

            this.secondary = value.ToLower();
        }
    }

    public Category(string primary, string secondary = null) {
        this.Primary = primary;
        if (secondary != null) {
        this.Secondary = secondary;
        }
    }

    public override string ToString() {
        if (string.IsNullOrEmpty(Secondary)) {
            return Primary;
        }
        return $"{Primary}/{Secondary}";
    }
}
```

Ohjelman ajo:

```
PS C:\Users\saket\tamk\OhSyTe\.NET\viikko1\EventTest> dotnet run
14.11.2023 .NET 8 released (microsoft/dotnet)
8.11.2022 .NET 7 released (microsoft)
8.11.2021 .NET 6 released (microsoft/test)

Sorted events:
8.11.2021 .NET 6 released (microsoft/test)
8.11.2022 .NET 7 released (microsoft)
14.11.2023 .NET 8 released (microsoft/dotnet)
```

Ajosta nähdään, että tulostus näyttää oikealta, kun annetaan secondary kategoria tai se jätetään antamatta. Kategoriat muutetaan myös automaattisesti pieniksi kirjaimiksi.

5.2 Tehtävä 10

Program.cs:

```
using System;
using System.Globalization;
using CsvHelper;

class Program {
    static void Main(string[] args) {
        // File path
        string filePath = "events.csv";

        // Creating events list
        List<Event> events = new List<Event>();

        // Reading events from events.csv
        using (var reader = new StreamReader(filePath))
        using (var csv = new CsvReader(reader, CultureInfo.InvariantCul-
ture)) {
            csv.Read();
            csv.ReadHeader();

            while (csv.Read()) {
                var date = csv.GetField<DateTime>("date");
                var description = csv.GetField<string>("description");
                var category = csv.GetField<string>("category");

                // Creating new event with every loop and adding it to
the list
                Event e = new Event(date.Date, description, category);
                events.Add(e);
            }
        }
```

```
        // Sorting events
        events.Sort();

        // Printing sorted events
        Console.WriteLine("\nSorted events:\n");
        foreach (var e in events) {
            Console.WriteLine(e);
        }
    }
}
```

Event.cs:

```
public class Event: IComparable {
    // Automatic property: direct access to private field
    public DateOnly Date { get; set; }
    // Private backing field; validation in setter
    private string description;
    private Category EventCategory;

    public string Description {
        get {
            return this.description;
        }

        set {
            // The incoming new value is always named 'value'.
            ArgumentNullException.ThrowIfNullOrEmpty(value);

            if (value.Length > 500) {
                throw new ArgumentException("Description is too long!");
            }
            this.description = value;
        }
    }

    public Event(DateTime date, string description, string primaryCate-
gory, string secondaryCategory = null) {
        this.Date = new DateOnly(date.Year, date.Month, date.Day);
        this.Description = description;
        this.EventCategory = new Category(primaryCategory, secondaryCate-
gory);
    }

    public override string ToString()
    {
        return $"{this.Date} {this.Description} ({this.EventCategory})";
    }
```

```
    // IComparable implementation
    public int CompareTo(object obj) {
        if (obj == null) {
            return 1;
        }

        Event otherEvent = obj as Event;
        if (otherEvent != null) {
            return this.Date.CompareTo(otherEvent.Date);
        }
        else {
            throw new ArgumentException("Object is not an event");
        }
    }
}
```

Category.cs:

```
public class Category {
    private string primary;
    private string secondary;

    public string Primary {
        get => this.primary;
        set {
            ArgumentNullException.ThrowIfNullOrEmpty(value);

            if (value.Length > 50) {
                throw new ArgumentException("Primary category is too
long!");
            }

            this.primary = value.ToLower();
        }
    }

    public string Secondary {
        get => this.secondary;
        set {
            ArgumentNullException.ThrowIfNullOrEmpty(value);

            if (value.Length > 50) {
                throw new ArgumentException("Secondary category is too
long!");
            }

            this.secondary = value.ToLower();
        }
```

```
    }

    public Category(string primary, string secondary = null) {
        this.Primary = primary;
        if (secondary != null) {
        this.Secondary = secondary;
        }
    }

    public override string ToString() {
        if (string.IsNullOrEmpty(Secondary)) {
            return Primary;
        }
        return $"{Primary}/{Secondary}";
    }
}
```

events.csv:

```
date,description,category
2024-02-15,C# and .NET basics,microsoft/dotnet
2024-02-08,Java strategy pattern,java
2023-11-14,.NET 8 released,microsoft/dotnet
2022-11-08,.NET 7 released,microsoft
2021-11-08,.NET 6 released,microsoft/TEST
2023-12-24,Christmas Eve,holidays
2023-12-25,Cristmas Day,holidays
2024-02-22,Ohjelmoinnin syventävät tekniikat,classes/ohsyte
2024-02-21,Data-analyysi ja tekoälyn perusteet,classes/datai
2020-12-15,C++20 released,c++/versions
```

Ohjelman ajo:

```
PS C:\Users\saket\tamk\OhSyTe\.NET\viikko1\EventTest> dotnet run

Sorted events:

15.12.2020 C++20 released (c++/versions)
8.11.2021 .NET 6 released (microsoft/test)
8.11.2022 .NET 7 released (microsoft)
14.11.2023 .NET 8 released (microsoft/dotnet)
24.12.2023 Christmas Eve (holidays)
25.12.2023 Cristmas Day (holidays)
8.2.2024 Java strategy pattern (java)
15.2.2024 C# and .NET basics (microsoft/dotnet)
21.2.2024 Data-analyysi ja tekoälyn perusteet (classes/datai)
22.2.2024 Ohjelmoinnin syventävät tekniikat (classes/ohsyte)
```

Ohjelma osaa lukea eventit tiedostosta oikein, järjestää ne ja tulostaa ne luokkien ToString() metodien mukaisesti.