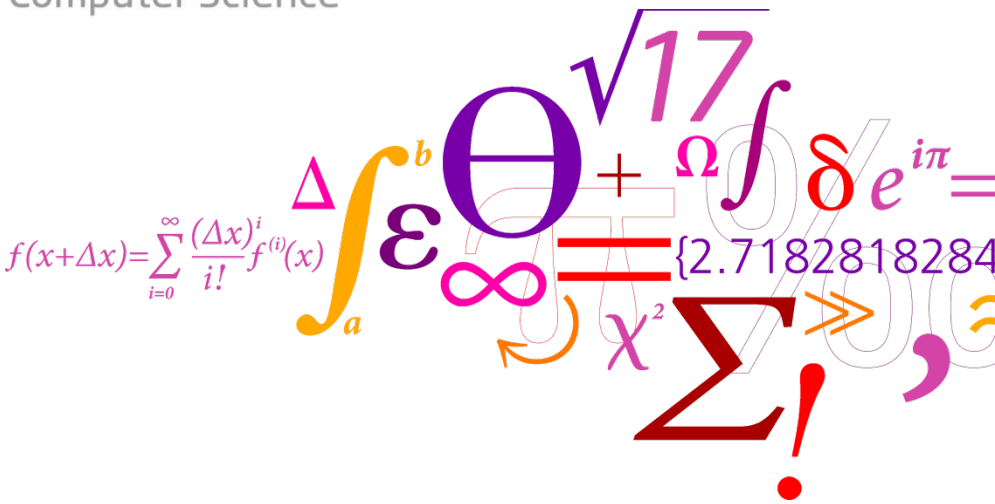
Projekt i software-udvikling (02362)
Forår 2022

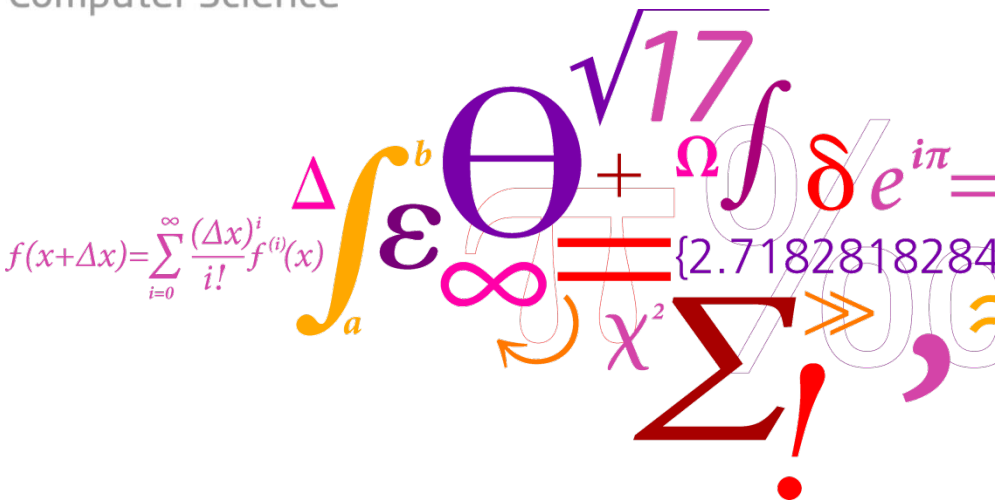
Ekkart Kindler

DTU Compute
Department of Applied Mathematics and Computer Science



IV. Java Praksis

DTU Compute
Department of Applied Mathematics and Computer Science



2. JavaFX Programmering

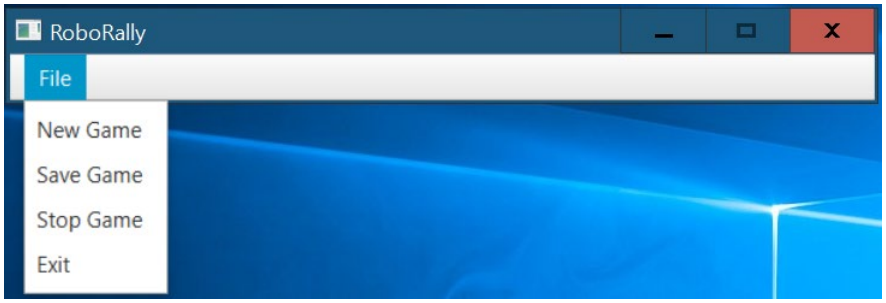
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

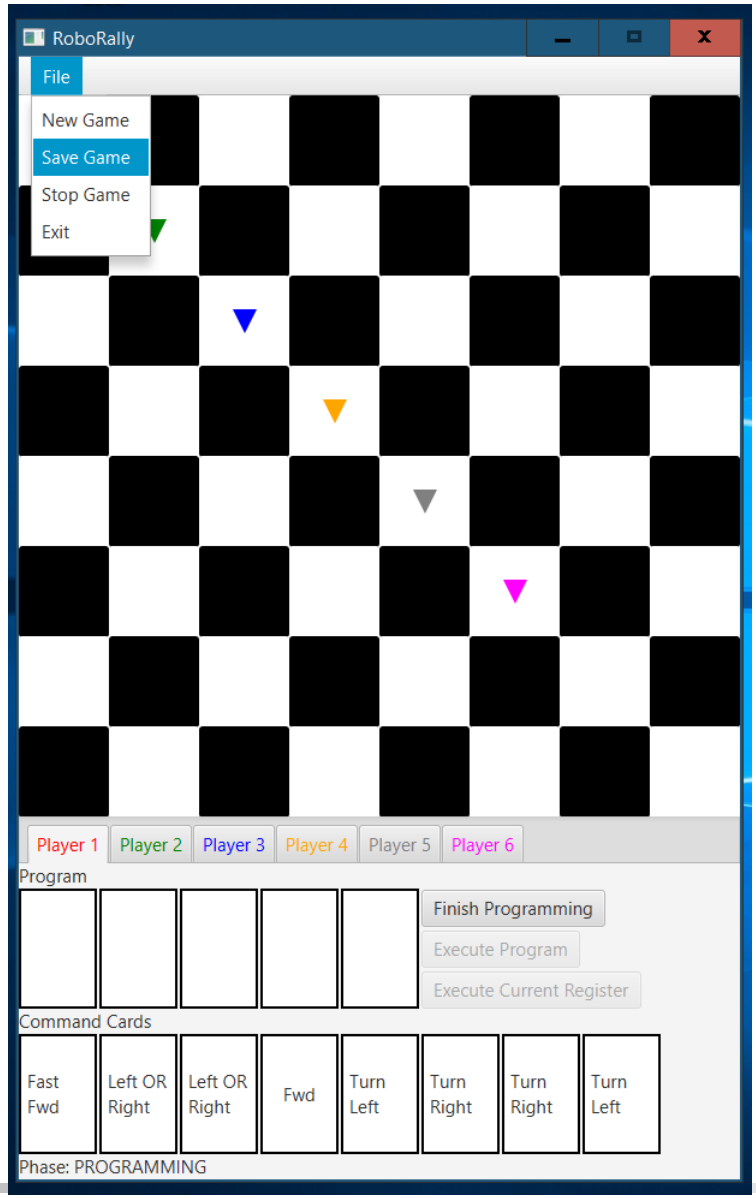
- Til at programmere GUIen burger vi JavaFX
- Her er der nogle informationer om JavaFX programmering:
- JavaFX Applikationen:
  - https://docs.oracle.com/javafx/2/get_started/hello_world.htm
  - https://docs.oracle.com/javafx/2/get_started/jfxpub-get_started.htm
- JavaFX LayoutPanes:
  - https://docs.oracle.com/javafx/2/layout/builtin_layouts.htm#
- JavaFX UI Controls:
  - https://docs.oracle.com/javafx/2/ui_controls/overview.htm#
  - https://docs.oracle.com/javafx/2/ui_controls/text-field.htm#
  - https://docs.oracle.com/javafx/2/ui_controls/button.htm#
- JavaFX Dialogs:
  - https://examples.javacodegeeks.com/desktop-java/javafx/dialog/javafx/javafx-dialog-example/

Fortsættes fra forelæsning 3

Menuer og menubjælker

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler





Kode følger på de næste slides.
Konceptuelt diskussion på tavlen
(notater følger på websiderne)

Eksempel: Menubjælke

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public class RoboRallyMenuBar extends MenuBar {

    private AppController appController;

    public RoboRallyMenuBar(AppController appController) {
        this.appController = appController;

        Menu controlMenu = new Menu("File");
        this.getMenus().add(controlMenu);

        MenuItem newGame = new MenuItem("New Game");
        newGame.setOnAction( e -> {this.appController.newGame();} );
        controlMenu.getItems().add(newGame);

        MenuItem saveGame = new MenuItem("Save Game");
        saveGame.setOnAction( e -> {this.appController.saveGame();} );
        controlMenu.getItems().add(saveGame);

        MenuItem stopGame = new MenuItem("Stop Game");
        stopGame.setOnAction( e -> {this.appController.stopGame();} );
        controlMenu.getItems().add(stopGame);

        MenuItem exitApp = new MenuItem("Exit
        exitApp.setOnAction( e -> {this.appContr
        controlMenu.getItems().add(exitApp);
    }

}
```

Menubjælke (MenuBar) importeres fra JavaFX

En controller for hele appen.

Fil-menu

Fil-menu tilføjes til bjælken

Menupunkt "New Game"

Det her er såkaldte λ-expressions (lambda), som hjælper at gøre koden mere læsbart (korter). Ikke i fokus her!

```
public class AppController {

    // ...

    private RoboRally roboRally;

    private GameController gameController;

    public AppController(RoboRally roboRally) {
        this.roboRally = roboRally;
    }
```

```
public void newGame() {
    ChoiceDialog<Integer> dialog =
      new ChoiceDialog<>(
        PLAYER_NUMBER_OPTIONS.get(0),
        PLAYER_NUMBER_OPTIONS);
    dialog.setTitle("Player number");
    dialog.setHeaderText("Select number");
    Optional<Integer> result = dialog.showAndWait();

    if (result.isPresent()) {
      // ...
      Board board = new Board(8, 8);
      gameController = new GameController(board);
```

```
            int no = result.get();
            for (int i = 0; i < no; i++) {
                Player player = new Player(
                  board,
                  PLAYER_COLORS.get(i),
                  "Player " + (i + 1));
                board.addPlayer(player);
                player.setSpace(board.getSpace(i, i));
            }
            board.setCurrentPlayer(board.getPlayer(0));
            roboRally.createBoardView(gameController);
            gameController.initializeProgrammingPhase();
        }
    }
    ...
```

Her er der også load() og save() som I skal bruge for at lade og gemme spil (opgave V4a).

Applikation

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public void start(Stage primaryStage) throws Exception {
    stage = primaryStage;

    AppController appController = new AppController(this);


    RoboRallyMenuBar menuBar = new
        RoboRallyMenuBar(appController);
    boardRoot = new BorderPane();
    VBox vbox = new VBox(menuBar, boardRoot);
    Scene primaryScene = new Scene(vbox);

    stage.setScene(primaryScene);
    stage.setTitle("RoboRally");
    stage.setResizable(false);
    stage.sizeToScene();
    stage.show();
}
```

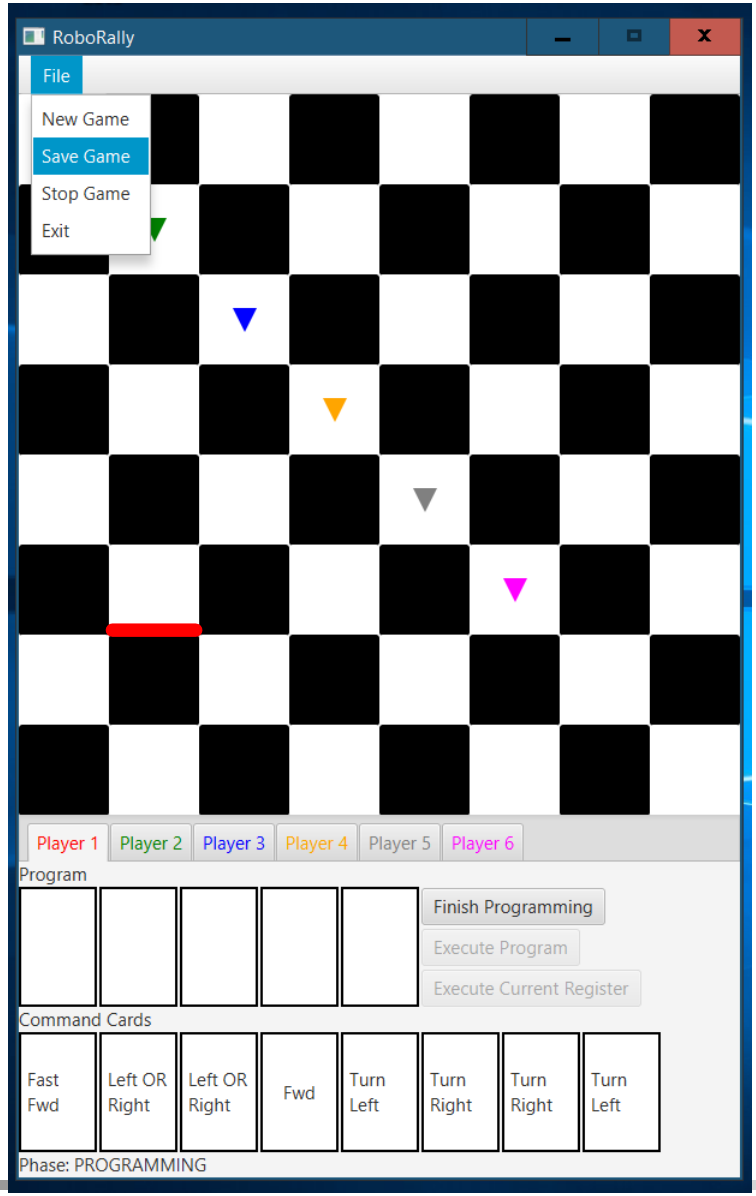
Applikationens AppController bliver kreeret og tilføjet til menubælken som også bliver kreeret.

```
public void createBoardView(GameController gameController) {
    // if present, remove old BoardView
    boardRoot.getChildren().clear();

    if (gameController != null) {
        // create and add view for new board
        BoardView boardView = new BoardView(gameController);
        boardRoot.setCenter(boardView);
    }

    stage.sizeToScene();
}
```

Sæt det nye BoardView på den forberedte plads.

Space med Væg

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



SpaceView

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Vi vil tegne et væg på den sydelige side (nederste side) af et felt i **`updateView()` eller `updatePlayer()`** metoden af klasse **`SpaceView`**.

Da **`SpaceView`** er (eller nedarver fra) et **`StackPane`**, bliver alle elemeneter som bliver tilføjet vist i midten (overfor hinanden): For at placerer vægge, er vi nødt til et kreere et element hvor vi kan placer andre objekter på. Her viser vi to muligheder hvordan man kan tegne vægge på et felt:

1. **`Pane`** hvor vi sætter linjer på
2. **`Canvas`** hvor vi tegner linjer på

SpaceView: Metode 1

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
Pane pane = new Pane();
Rectangle rectangle =
  new Rectangle(0.0, 0.0, SPACE_WIDTH, SPACE_HEIGHT);
rectangle.setFill(Color.TRANSPARENT);
pane.getChildren().add(rectangle);

// SOUTH
Line line =
  new Line(2, SPACE_HEIGHT-2,
           SPACE_WIDTH-2, SPACE_HEIGHT-2);
line.setStroke(Color.RED);
line.setStrokeWidth(5);
pane.getChildren().add(line);

this.getChildren().add(pane);
```

SpaceView: Metode 2

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
Canvas canvas =
   new Canvas(SPACE_WIDTH, SPACE_HEIGHT);

GraphicsContext gc =
   canvas.getGraphicsContext2D();
gc.setStroke(Color.RED);
gc.setLineWidth(5);
gc.setLineCap(StrokeLineCap.ROUND);

gc.strokeLine(2, SPACE_HEIGHT-2,
   SPACE_WIDTH-2, SPACE_HEIGHT-2);

this.getChildren().add(canvas);
```

4. Undtagelser (Exceptions)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Nogle gange kommer det frem midt i en metode, at operationen ikke kan gennemføres. Så er man nødt til at reagere og også den som har kaldt metoden er måske nødt til at reagere på det.

Såkaldte undtagelser (exceptions) er en mulighed at programmere sådan noget.

Eksempel: Flyt en robot

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Hvis en robot skal flyttes, skubber den en robot som star foran den også I same retning; og hvis der star en foran den igen, skal den også flyttes, osv.
- Det giver en kaskade af flytningerne
- Hvis så den sidste i kaskaden ikke kan flyttes (fx da den rammer en væg), så ryger hele kaskaden, og den oprindelige træk er ikke muligt

→ Se diskussion på tavlen!

Deklaration af Exception

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
class ImpossibleMoveException extends Exception {

    private Player player;
    private Space space;
    private Heading heading;

    public ImpossibleMoveException(Player player,
            Space space,
            Heading heading) {
        super("Move impossible");
        this.player = player;
        this.space = space;
        this.heading = heading;
    }
}
```

MoveForward()

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public void moveForward(@NotNull Player player) {
    if (player.board == board) {
        Space space = player.getSpace();
        Heading heading = player.getHeading();

        Space target = board.getNeighbour(space, heading);
        if (target != null) {
            try {
                moveToSpace(player, target, heading);
            } catch (ImpossibleMoveException e) {
                // we don't do anything here  for now;
                // we just catch the exception so that
                // we do no pass it on to the caller
                // (which would be very bad style).
}   }   }   }
```

Skubbe-metoden

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
private void moveToSpace(
        @NotNull Player player,
        @NotNull Space space,
        @NotNull Heading heading) throws ImpossibleMoveException {

    Player other = space.getPlayer();
    if (other != null){
        Space target = board.getNeighbour(space, heading);
        if (target != null) {
            // XXX Note that there might be additional problems
            //     with infinite recursion here!
            moveToSpace(other, target, heading);
        } else {
            throw new ImpossibleMoveException(player, space, heading);
        }
    }
    player.setSpace(space);
}
```

Disse to metoder er med i den version af softwaren i får udleveret i dag: roborally-1.4.1.

5. JDBC: Prepared Statements

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Prepared statements er en nem (og lidt mindre fejlbehæftede) metode for at interagere med en database
- Man kan bruge SELECT-statements til at **opdatere tabeller**
- I min kode har jeg kun et eneste INSERT-Statement og ingen UPDATE-statements; alle andre er SELECT-statements, som også bliver brugt til at opdatere og endda til at tilføje data til databasen (→ slide 31)

Den viste kode og lidt mere (også til næste uge) bliver udleveret som partiel IntelliJ-projekt: roborally-1.4.1.

I kan kopiere kode derfra til jeres nuværende løsning.

Connector

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
class Connector {

    private static final String HOST     = "localhost";
    private static final int    PORT     = 3306;
    private static final String DATABASE = "pisu";
    private static final String USERNAME = "root";
    private static final String PASSWORD = "";

    private static final String DELIMITER = ";;";

    private Connection connection;

    Connector() {
        try {
         String url = "jdbc:mysql://" + HOST + ":" + PORT + "/" +
                      DATABASE + "?serverTimezone=UTC";
         connection = DriverManager.getConnection(url, USERNAME, PASSWORD

         createDatabaseSchema();
      } catch (SQLException e) {
         // TODO we should try to diagnose and fix some problems here and
         //      exit in a more graceful way
         e.printStackTrace();
      }
    }
```

Connector …

```
private void createDatabaseSchema() {

    String createTablesStatement;
    try {
        ClassLoader classLoader = Connector.class.getClassLoader();
        URI uri = classLoader.
             getResource("schemas/createschema.sql").toURI();
        byte[] bytes = Files.readAllBytes(Paths.get(uri));
        createTablesStatement = new String(bytes);
    } catch (URISyntaxException | IOException e) {
        e.printStackTrace();
        return;
    }

    try {
        connection.setAutoCommit(false);

        Statement statement = connection.createStatement();
        for (String sql : createTablesStatement.split(DELIMITER)) {
            if (!StringUtils.isEmptyOrWhitespaceOnly(sql)) {
                statement.executeUpdate(sql);
            }
        }
        statement.close();

        connection.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        // TODO error handling
        try {
          connection.rollback();
        } catch (SQLException e1) {}
    } finally {
        try {
          connection.setAutoCommit(true);
        } catch (SQLException e) {}
    }
}
```

Koden sørger for at tabellerne bliver automatisk kreeret i databasen, hvis de ikke eksisterer i forvejen! Dertil læses skemate fra "createschema.sql" resourcen.

Det hele skal foregår transaktionelt (helt eller ingenting).

Hovedidee

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
ClassLoader classLoader =
  Connector.class.getClassLoader();
URI uri = classLoader.
  getResource("schemas/createschema.sql").toURI();

byte[] bytes = Files.readAllBytes(Paths.get(uri));
String createTablesStatement = new String(bytes);

connection.setAutoCommit(false);
Statement statement = connection.createStatement();
for (String sql :
    createTablesStatement.split(DELIMITER)) {
  if (!StringUtils.isEmptyOrWhitespaceOnly(sql)) {
    statement.executeUpdate(sql);
} }
statement.close();
connection.commit();
```

```
private void createDatabaseSchema() {

    String createTablesStatement;
    try {
      ClassLoader classLoader =
            Connector.class.getClassLoader();
      URI uri = classLoader.
            getResource("schemas/createschema.sql").toURI();
       byte[] bytes = Files.readAllBytes(Paths.get(uri));
       createTablesStatement = new String(bytes);
    } catch (URISyntaxException | IOException e) {
       e.printStackTrace();
       return;
    }

    // ...
```

```
    try {
        connection.setAutoCommit(false);
        Statement statement = connection.createStatement();
        for (String sql : createTablesStatement.split(DELIMITER)
            if (!StringUtils.isEmptyOrWhitespaceOnly(sql)) {
                statement.executeUpdate(sql);
            }
        }

        statement.close();
        connection.commit();
    } catch (SQLException e) {
        e.printStackTrace();
        // TODO error handling
        try {
          connection.rollback();
      } catch (SQLException e1) {}
    } finally {
      try {
          connection.setAutoCommit(true);
      } catch (SQLException e) {}
    }
}
```

SQL (Excerpt) …

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
CREATE TABLE IF NOT EXISTS Game (

  gameID int NOT NULL UNIQUE AUTO_INCREMENT,

  name varchar(255),

  currentPlayer tinyint NULL,
  phase tinyint,
  step tinyint,

  PRIMARY KEY (gameID),
  ...

);;
```

Her er et udtræk af "createschema.sql" resourcen.

SQL (Excerpt)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
CREATE TABLE IF NOT EXISTS Player (

  gameID int NOT NULL,
  playerID tinyint NOT NULL,

  name varchar(255),
  colour varchar(31),

  positionX int,
  positionY int,
  heading tinyint,

  PRIMARY KEY (gameID, playerID),
  FOREIGN KEY (gameID) REFERENCES Game(gameID)
  ...

);;

...
```

DAL: Interface

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public interface IRepository {

    boolean createGameInDB(Board game);

    boolean updateGameInDB(Board game);

    Board loadGameFromDB(int id);

    List<GameInDB> getGames();

}
```

Det her er en meget brutal og uelegant løsning: Vi henter en liste med alle spil og deres numre fra databasen.

DAL: Implementering

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
class Repository implements IRepository {

   private static final String GAME_GAMEID = "gameID";

   private static final String GAME_NAME = "name";

   private static final String GAME_CURRENTPLAYER =
      "currentPlayer";

   private static final String GAME_PHASE = "phase";

   private static final String GAME_STEP = "step";

   ...
```

```
public boolean createGameInDB(Board game) {
    if (game.getGameId() == null) {
        Connection connection = connector.getConnection();
        try {
            connection.setAutoCommit(false);

            PreparedStatement ps = getInsertGameStatementRGK();
            ps.setString(1, "Date: " +  new Date());
            ps.setNull(2, Types.TINYINT); //
            ps.setInt(3, game.getPhase().ordinal());
            ps.setInt(4, game.getStep());


            int affectedRows = ps.executeUpdate();
            ResultSet generatedKeys = ps.getGeneratedKeys();
            if (affectedRows == 1 && generatedKeys.next()) {
                game.setGameId(generatedKeys.getInt(1));
            }
            generatedKeys.close();

            createPlayersInDB(game);
            createCardFieldsInDB(game);
```

Den aktuelle spiller bliver sat til NULL for nu!

Se `getInsertGameStatementRGK()` for rækkefølgen (→ slide 32).

Den kode er med i roborally-1.4.1

Den kode er **ikke** med i roborally-1.4.1

```
        ps = getSelectGameStatementU();
        ps.setInt(1, game.getGameId());

        ResultSet rs = ps.executeQuery();
        if (rs.next()) {
            rs.updateInt(GAME_CURRENTPLAYER,
              game.getPlayerNumber(game.getCurrentPlayer()));
            rs.updateRow();
        } else {
            // TODO error handling
        }
        rs.close();
        connection.commit();
        connection.setAutoCommit(true);
        return true;
    } catch (SQLException e) {
        // TODO error handling
        try {
            connection.rollback();
            connection.setAutoCommit(true);
        } catch (SQLException e1) {
            // TODO error handling
            e1.printStackTrace();
        }
    }
} else {
    System.err.println("Game cannot be created in DB, " +
      "since it has a game id already!");
```

Se slide 33!

Her bliver den aktuelle spiller endelig opdateret! Hvorfor først her?

Alle ændringer bliver først gennemført i databasen her ..

… eller hvis der slog noget fejl rullet tilbage her!

```
private static final String SQL_INSERT_GAME =
      "INSERT INTO Game(name, currentPlayer, phase, step) " +
        "VALUES (?, ?, ?, ?)";

private PreparedStatement insert_game_stmt = null;

private PreparedStatement getInsertGameStatementRGK() {
   if (insert_game_stmt == null) {
      Connection connection = connector.getConnection();
      try {
         insert_game_stmt = connection.prepareStatement(
               SQL_INSERT_GAME,
               Statement.RETURN_GENERATED_KEYS);
      } catch (SQLException e) {
         // TODO error handling
         e.printStackTrace();
      }
   }
   return insert_game_stmt;
}
```

Da tabellen Game har automatisk genererede nøgler, er vi interesseret i at få denne nøgle og tilføje det til Board-objektet, så snart vi tilknytter det til databasen: → slide 26 og 30

```
private static final String SQL_SELECT_GAME =
        "SELECT * FROM Game WHERE gameID = ?";

private PreparedStatement select_game_stmt = null;

private PreparedStatement getSelectGameStatementU() {
    if (select_game_stmt == null) {
        Connection connection = connector.getConnection();
        try {
            select_game_stmt = connection.prepareStatement(
                    SQL_SELECT_GAME,
                    ResultSet.TYPE_FORWARD_ONLY,
                    ResultSet.CONCUR_UPDATABLE);
        } catch (SQLException e) {
            // TODO error handling
            e.printStackTrace();
        }
    }
    return select_game_stmt;
}
```

Bemærk at dette SELECT-statement bliver brugt til at **opdatere** game-tabellen (→ slides 26 og 31).

Documentation

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

JavaDocs JDBC Basics

https://docs.oracle.com/javase/tutorial/jdbc/basics/index.html

Java Doc JDBC: Using Prepared Statements

https://docs.oracle.com/javase/tutorial/jdbc/basics/prepared.html

Java Doc JDBC: Retrieving and Modifying Values from Results Sets:

https://docs.oracle.com/javase/tutorial/jdbc/basics/retrieving.html

Se nøgleord:

- **`ResultsSet`**
- Cursor
- Updating Rows in **`ResultSet`**
- Inserting Row in **`ResultSet`**
- **`CONCUR_UPDATABLE`**

Læs op til næste uge!

Diskuteres lidt mere næste uge!

Opgave V4a (til 29. 3.)

Det er først om 3 uger!

Jeres software skal kunne **gemme og genlade spil i/fra en database** (visnings opgave):

- spillets tilstand,
- spillerens/robottens tilstand inkl.
- spillerens kort (på hånden) og især programmet

Når spillet genlades, skal også GUIen vise den aktuelle situation rigtig (især når den var interaktiv modus). Men hvis MVC og AppControler er sat op hensigtsmæssigt, burde det næsten være ”gratis”.

Vi diskuterer mere næste uge (og der udvides også opgaven så at en spilleplade kan lades V4b)

Opgave V4a

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Koden som blev diskuteret i dag (og lidt mere som også dækker over opgave V4b) får I udleveret som IntelliJ-projekt roborally-1.4.1!
- Dette projekt er ikke komplet. Men I kan kopiere dele af koden til jeres nuværende projekt og bruge det og i jeres **`AppController`** for at gemme og genlade spil (via interface **`IRepository`** og **`RepositoryAccess`**).
- Bemærk at denne version kun gemmer spillet og dettes spillere, men ikke spillernes kort. Denne del skal I selv tilføje!

Opgave V4a

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Find de rigtige steder i jeres **`AppController`** hvor metoder af **`IRepository`** skal bruges/kaldes.

- På langt sigt kunne det også tænkes at et spil bliver automatisk gemt efter hvert træk; men det er ikke en del af Opgave V4a.

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Motivation

- Implementeringer i programmer er ofte meget teknisk
- Der er mange data som kan ødelægges, når man ikke bruger dem rigtigt
- Man ved ikke hvad der er vigtigt og ikke så vigtigt

- Programmet er svært at forstå
- Programmet kan nemt ødelægges

Motivation

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Vi har brug for mechanismer som hjælper os med at

- fokussere på de vigtige ting:

  Abstraktion

- Gemmer implementeringsdetailjer

  Hiding

Eksempel: Interface `Stack`

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

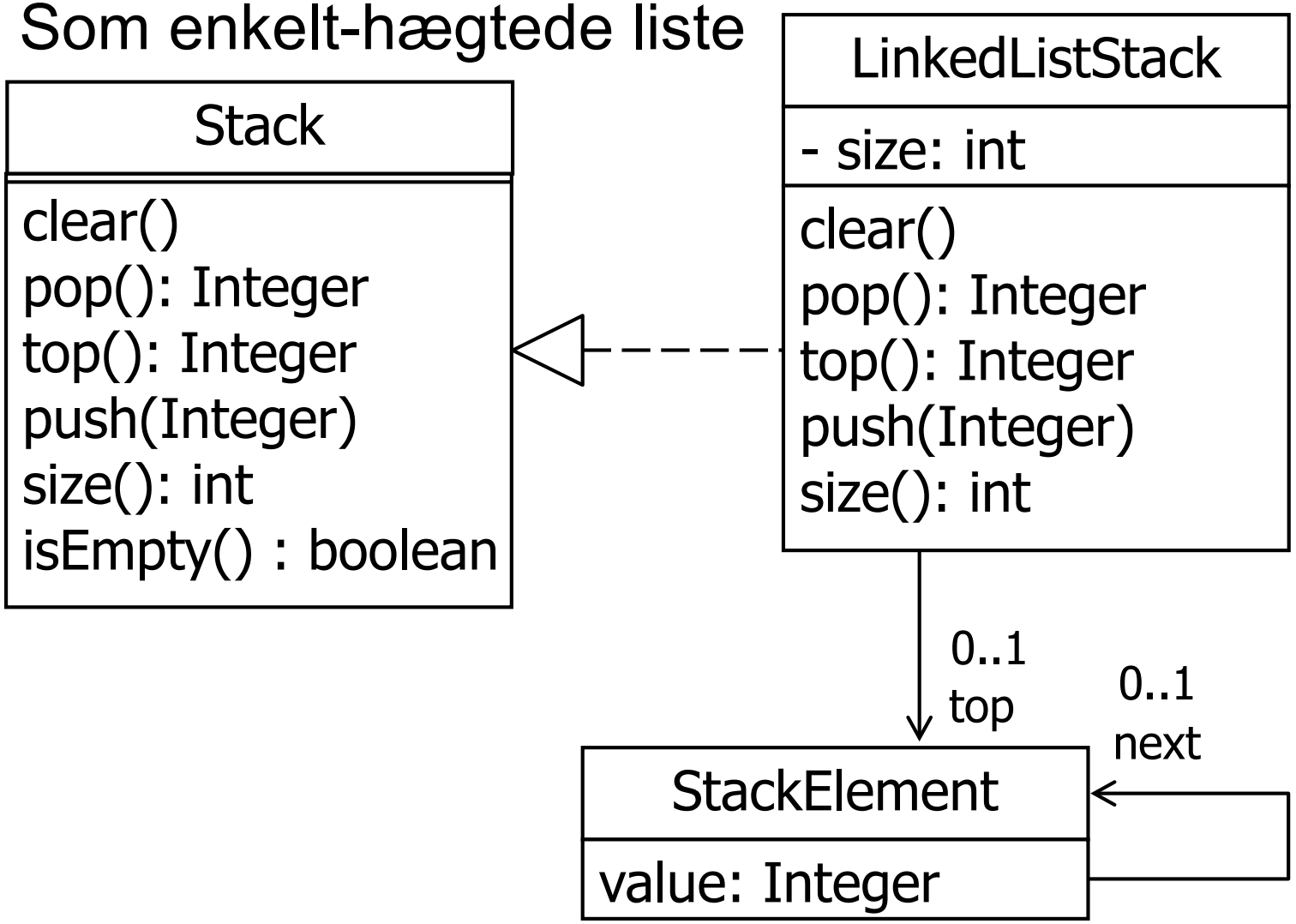
```
public interface Stack {

  void clear();

  Integer pop();

  Integer top();

  void push(Integer value);

  int size();

  default boolean isEmpty() {
    return size() == 0;
  }
}
```

En stak af heltals-objekter (`Integer`)

Implementering

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Som enkelt-hægtede liste



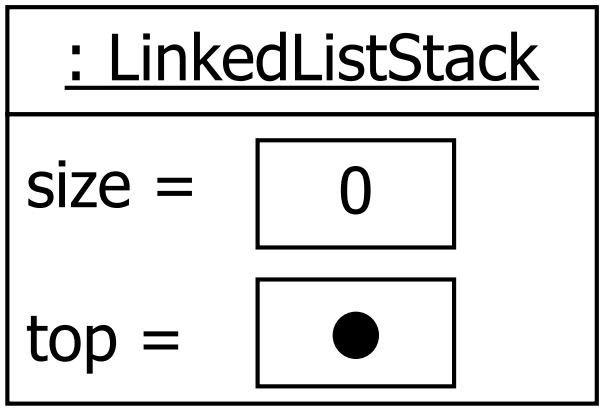
```
public class LinkedListStack implements Stack {

   private StackElement top = null;

   private int size = 0;

   public void clear() {
      top = null;
      size = 0;
   }

   public Integer pop() {
      if (top != null) {
         StackElement element = top;
         top = element.next;
         size--;
         return element.value;
      }

      return null;
   }
```

```
public Integer top() {
    if (top != null) {
        return top.value;
    }

    return null;
}

public void push(Integer value) {
    StackElement newElement = new StackElement(value, top);
    size++;
    top = newElement;
}

public int size() {
    return size;
}
```

```
    private class StackElement {

        private Integer value;

        private StackElement next;

        private StackElement(Integer value, StackElement next) {
            this.value = value;
            this.next = next;
        }
    }

}
```

Instans (”Objektdiagram”)

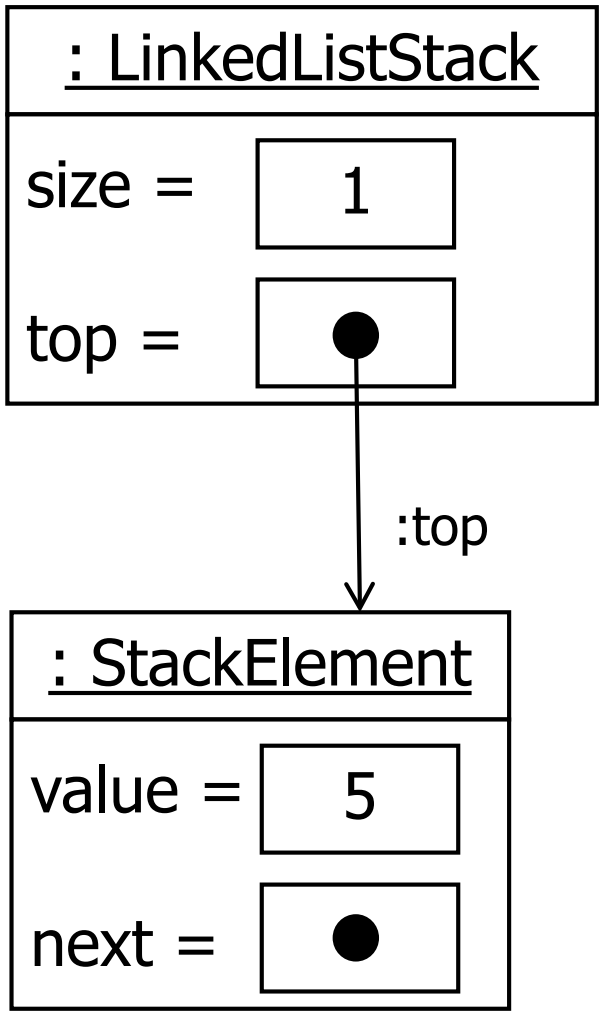
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Tom stak

Instance ("Objectdiagram")

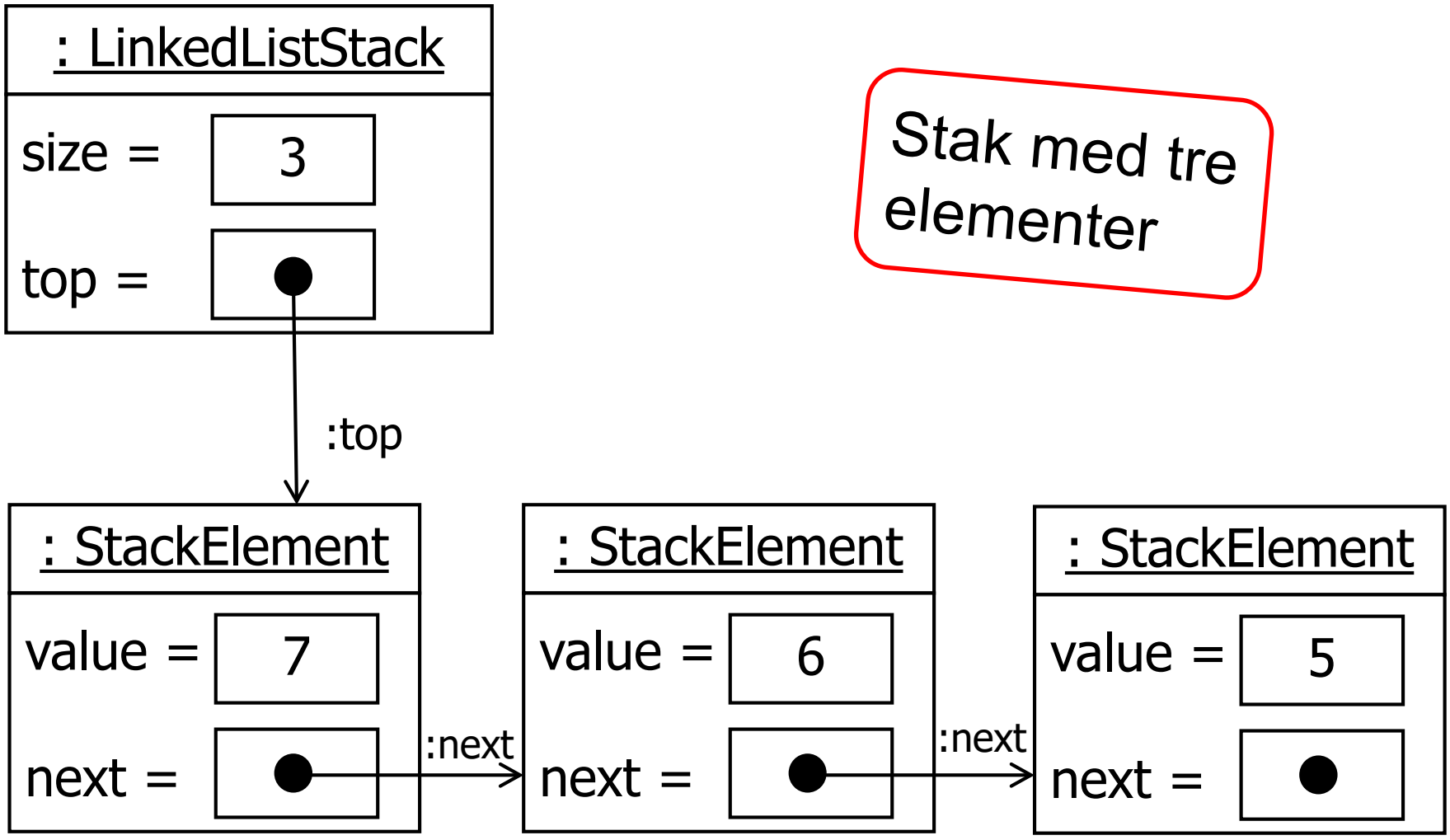
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



Stak med et element

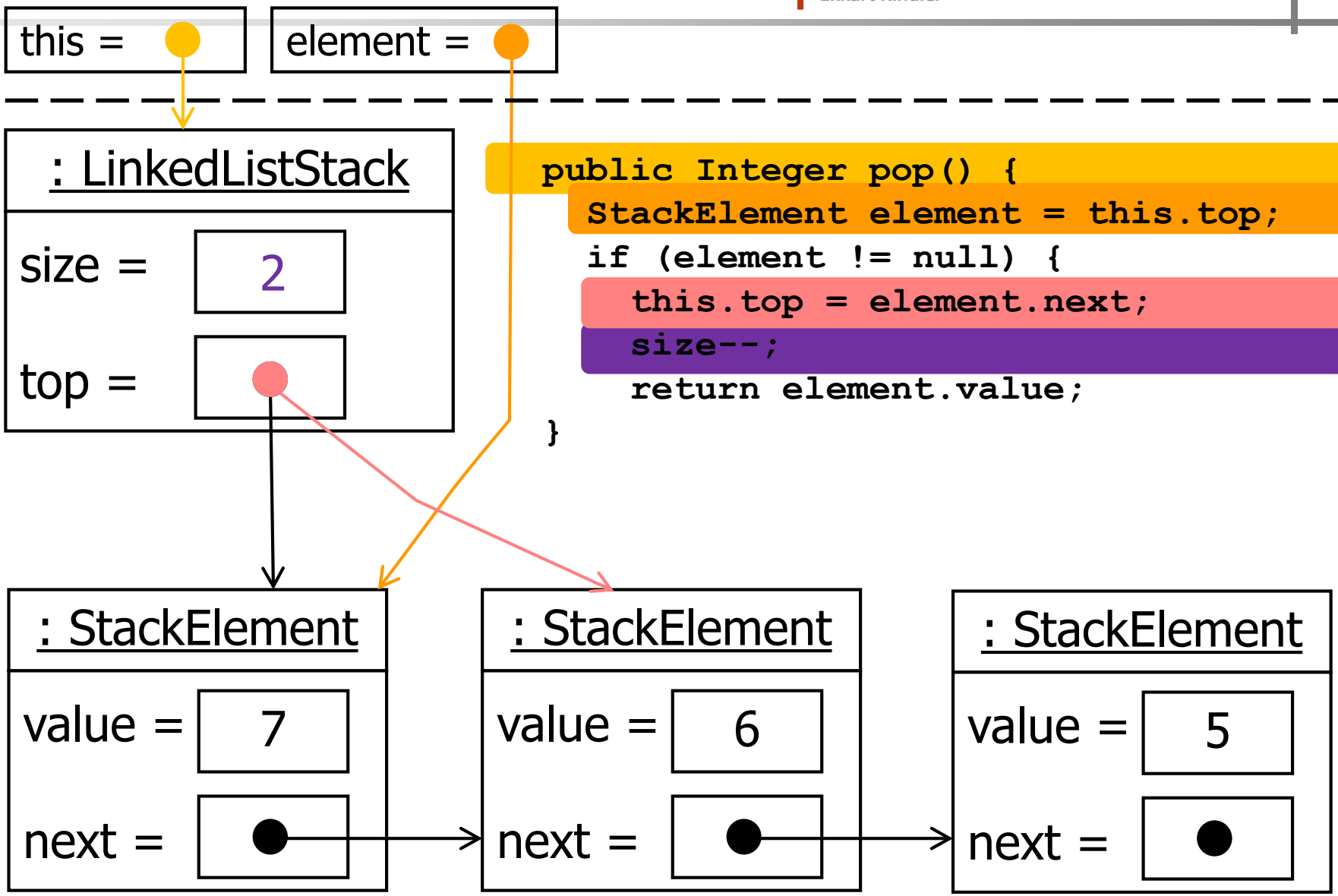
Instance ("Objectdiagram")

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



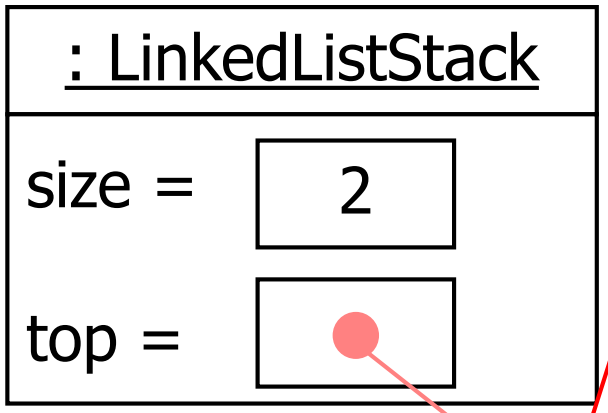
pop()

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

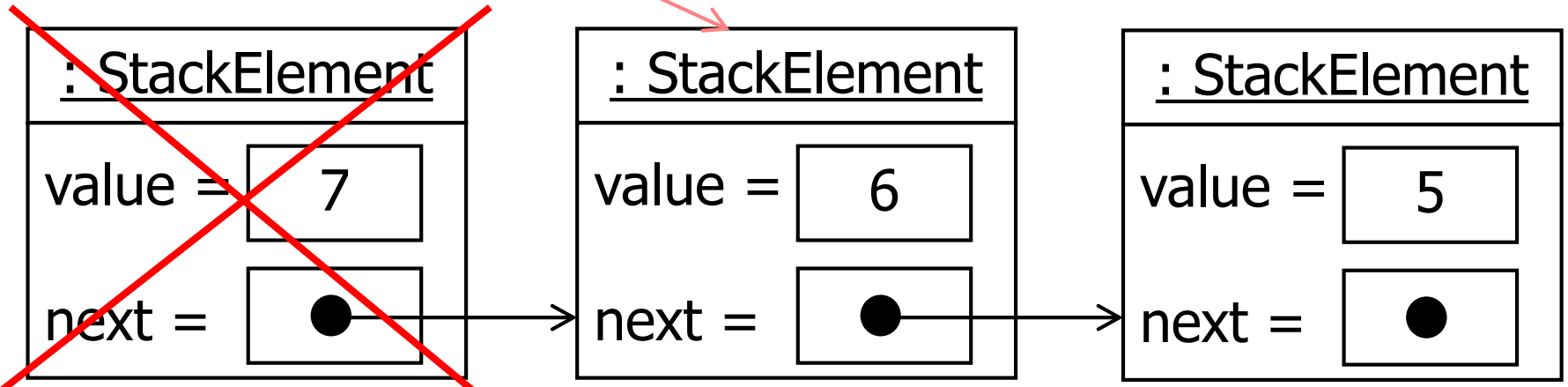


pop()

Dette objekt vil blive "garbage collected senere"

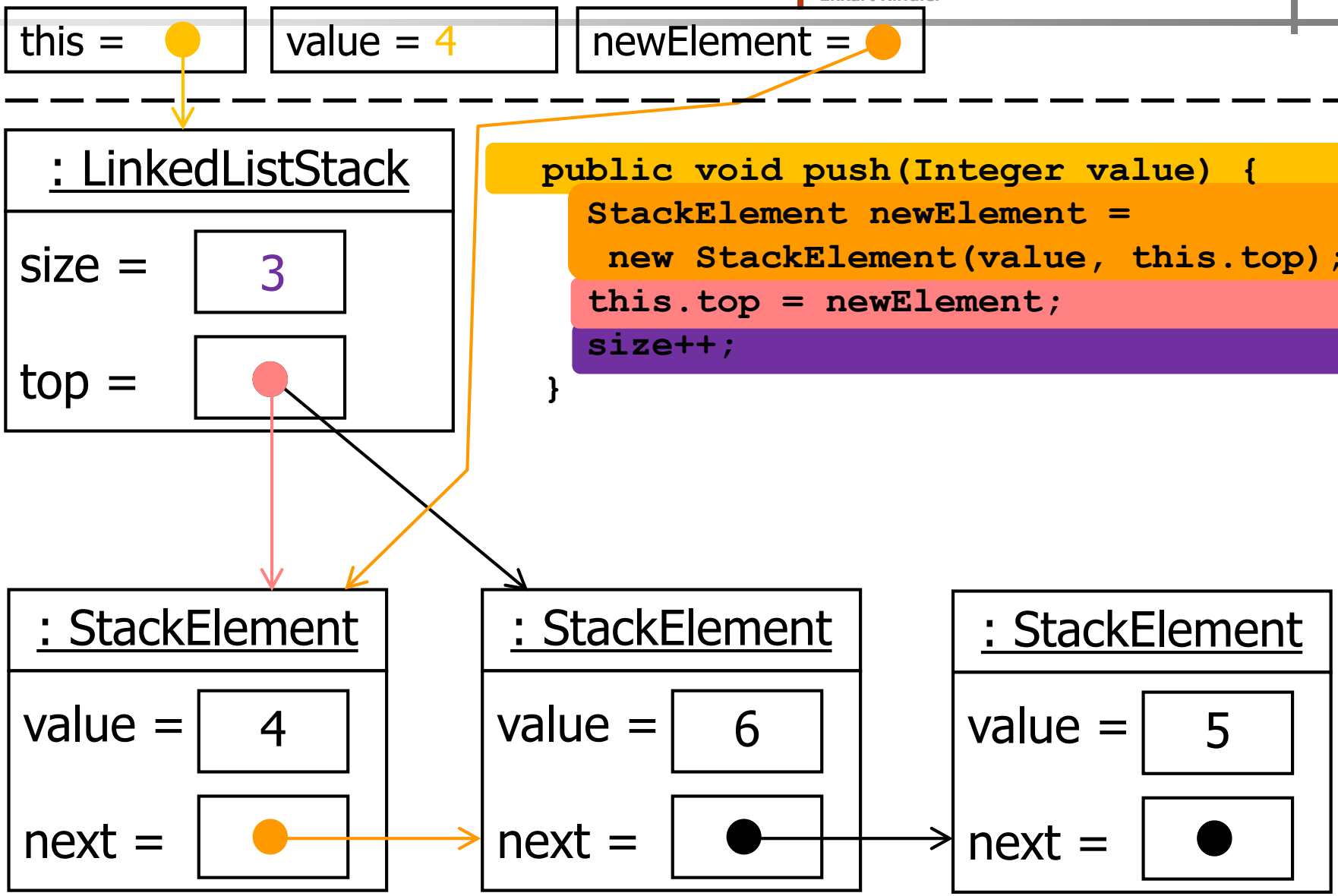



```
lic Integer pop() {
StackElement element = this.top;
if (element != null) {
   this.top = element.next;
   size--;
   return element.value;
}
```



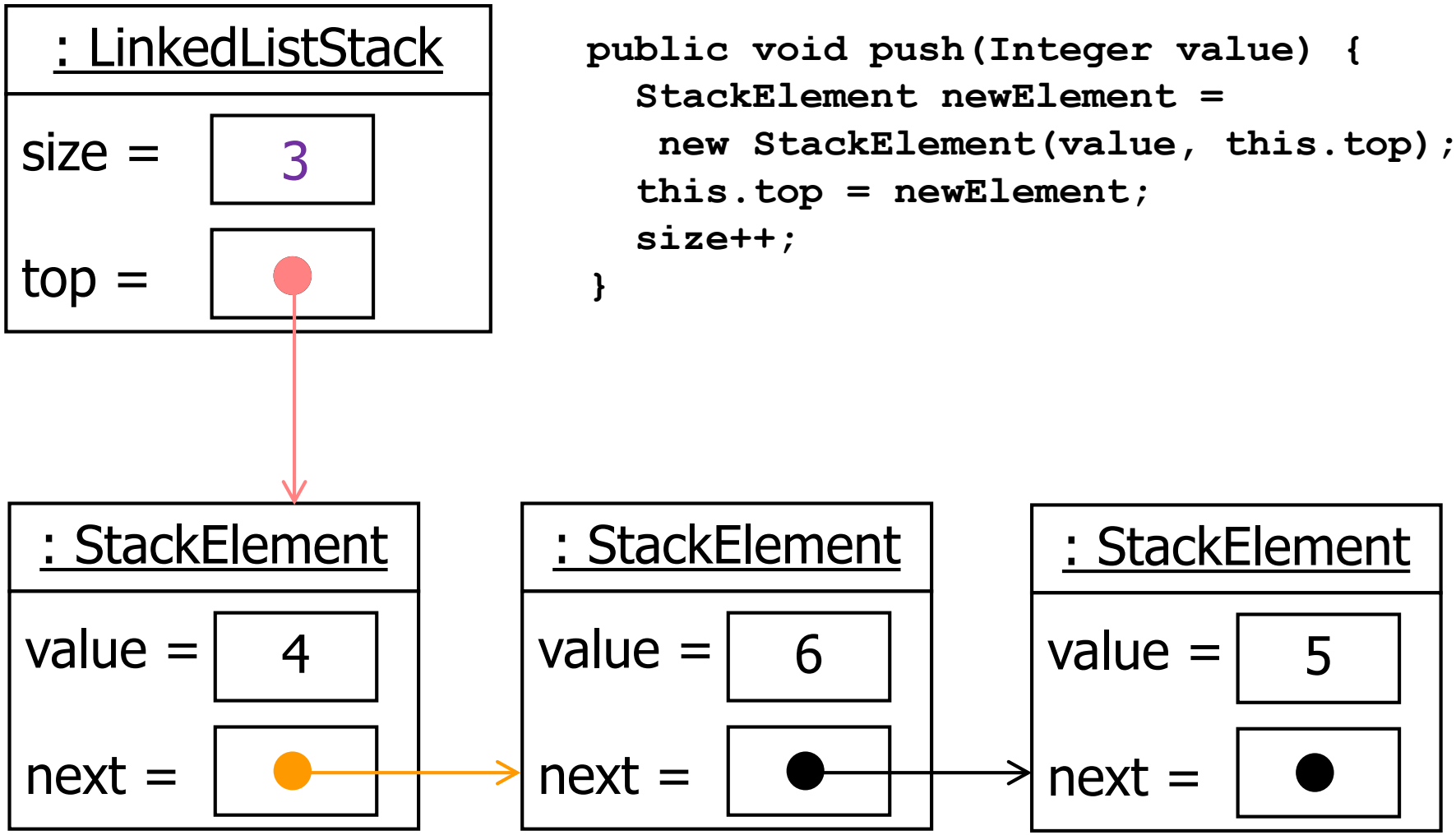
push()

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



push()

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler




```
public void push(Integer value) {
   StackElement newElement =
    new StackElement(value, this.top);
   this.top = newElement;
   size++;
}
```

Generisk `Stack`

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public interface Stack<E> {

  void clear();

  E pop();

  E top();

  void push(E value);

  int size();

  default boolean isEmpty() {
    return size() == 0;
  }
}
```

Lige så nemt kunne man definere og realisere en stak med elementer af hvilken som helst type `E`: generisk stak: Bare erstat `Integer` med `E`.

Spørgsmål

Hvordan kan adskille ægte resultater fra resultater, som er en slags fejlmeldinger?

Svar: Exceptions

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

Metoder kan ”kaste” (engl. *throw*) undtagelser (engl. *exceptions)*.

Dem som har kaldt metoden som kaster (engl. *throw*) en *exception*, kan enten

- fange (engl. catch) denne exception og fortsætte med beregningen eller
- ikke fange den, så ryger exception op et niveau højere op (i dette tilfælde kan metoden ikke fortsætte selv – undtaget **`finally`**, som vi diskuterer senere)

Hvis ingen fanger exception, så stopper hele programmet/beregning med denne exception.

Eks.: Stak (interface)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public interface Stack<E> {

  void clear();

  E pop() throws IllegalStateException;

  E top() throws IllegalStateException;

  void push(E value) throws IllegalArgumentException;

  ...

  int size();

}
```

throws betyder kun: metoden "kan kaste", men ikke at metoden nødvendigvis gør det!

Vi definerer her at de forskellige metoder kan kaste forskellige exceptions. De exceptions vi bruger her er Javas indbyggede exceptions. Men man kan også definere egne exceptions (se vores eksempel fra sidste uge).

Eks.: Stak (implement.)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public class LinkedListStack<E> implements Stack<E> {

    ...

    public E pop() throws IllegalStateException {
        if (top != null) {
            StackElement<E> element = top;
            top = element.next;
            size--;
            return element.value;
        }
        throw new IllegalStateException();
    }

    ...
```

throw betyder at en exception bliver kastet her!

Eks.: Stak (implement.)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public void push(E value)
   throws IllegalArgumentException {
        StackElement<E> newElement =
              new StackElement<>(value, top);
        size++;
        top = newElement;
}
```

Implementeringen af `push()` kaster ikke en exception, men `new StackElement()` gør (se næste slide); og da `push()` ikke fanger den ryger den videre op til kalderen af `push()`.

Eks.: Stak (implement.)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
...
private class StackElement<E> {
        private E value;
        private StackElement next;

        private StackElement<E>(
                        E value,
                        StackElement<E> next)
                    throws IllegalArgumentException {
            if (value == null) {
              throw new IllegalArgumentException();
            }
            this.value = value;
            this.next = next;
} } }
```

Eks.: Stak (brug)

Vi bruger `Integer` nu for type-parameter `E`; så har vi en stak til heltal objekter igen.

```
public void test() {
  Stack<Integer> stack = new LinkedListStack<Integer>();
  ...
  stack.clear();
  stack.pop();
}
```

`pop()` udløser (kaster) en `IllegalStateException`, men da den ikke bliver fanget ryger den op til den som kaldte `test()` metoden.

Eks.: Stak (brug)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public void test() {
   Stack<Integer> stack = new LinkedListStack<Integer>();
   ...
   stack.clear();
   try {
      stack.pop();
   } catch (IllegalStateException e) {
      // do something to fix things
   }
}
```

I det her eksempel ville det selvfølgelig være bedre at tjekke om stakken er tom (`isEmpty()`) inden man kalder `pop()`.

`pop()` udløser (kaster) en `IllegalStateException`, den bliver fanget af catch blokken. Derfor ryger den ikke videre op. I catch blokken kan man prøve at oprette om problemet. Man kan også tilgå exception ved brug af variable `e`,

Eks.: Stak (brug)

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

```
public void test() {
  Stack<Integer> stack = new LinkedListStack<Integer>();
  ...

  try {
    stack.pop();
  } catch (IllegalStateException e) {
    // do something to fix things
  } finally {
    System.out.println("This will always be printed!");
  }
}
```

`finally` blokken bliver altid gennemført, hvis try blokken bliver startet – uanset om der bliver fanget en exception eller ej. Det kan man bruge til at ryde op.

Specificere Exceptions

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Indtil videre har vi specificeret alle exceptions som en metode kan kaste
- Nogle gange bliver det for meget at specificere alle exceptions, som en metode kan kaste. Det gælder især exceptions som man ikke kan gøre noget ved
- Derfor er der to slags exceptions i Java:
  - *Overvågede* (checked) exceptions
  - *Uovervågede* (unchecked) exceptions

Overvågede exceptions

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- er exceptions, hvor den, som kalder metoden (*engl. caller*), kan gøre noget ved

  typisk: forkert brug af metoden (kald den med de rigtige parameter), eller IOException (prøv igen) og tilgang til database (se senere)

- skal specificeres I metodens deklaration eller fanges I metodens implementering

Uovervågede exceptions

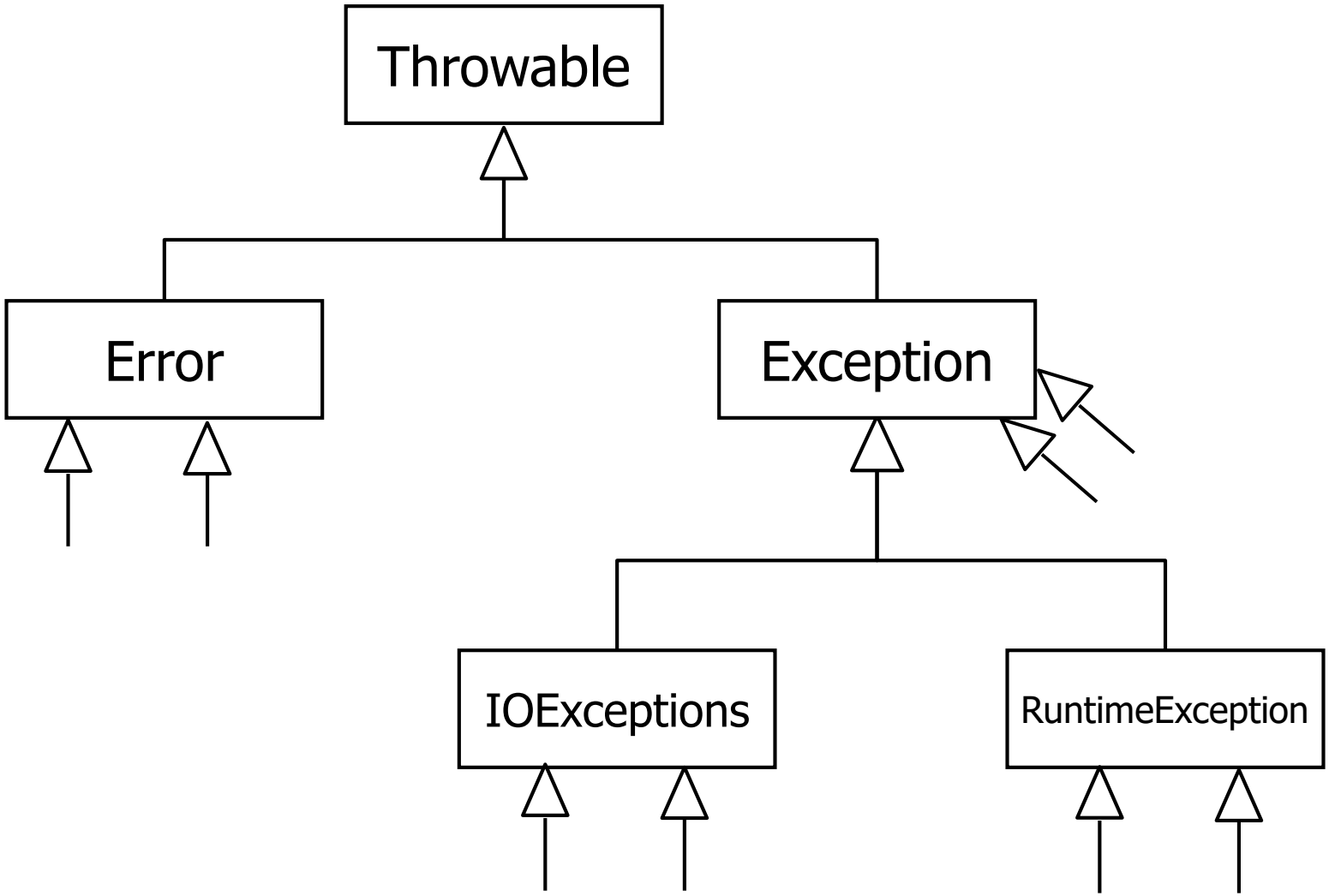
DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- er exceptions, hvor den, som kalder metoden (*engl. caller*), ikke kan gøre noget ved
  typisk: programmeringsfejl i implementering
  (NullPointerException, ClassCastException, ArrayIndexOutOfBounds, StackOverflowError, IOError, …)

- er man ikke nødt til at specificere I metodens deklaration, selvom de bliver kastet eller ikke fanget I metodens implementering
- ryger typisk op til main-tråden og stopper hele programmet, når de optræder

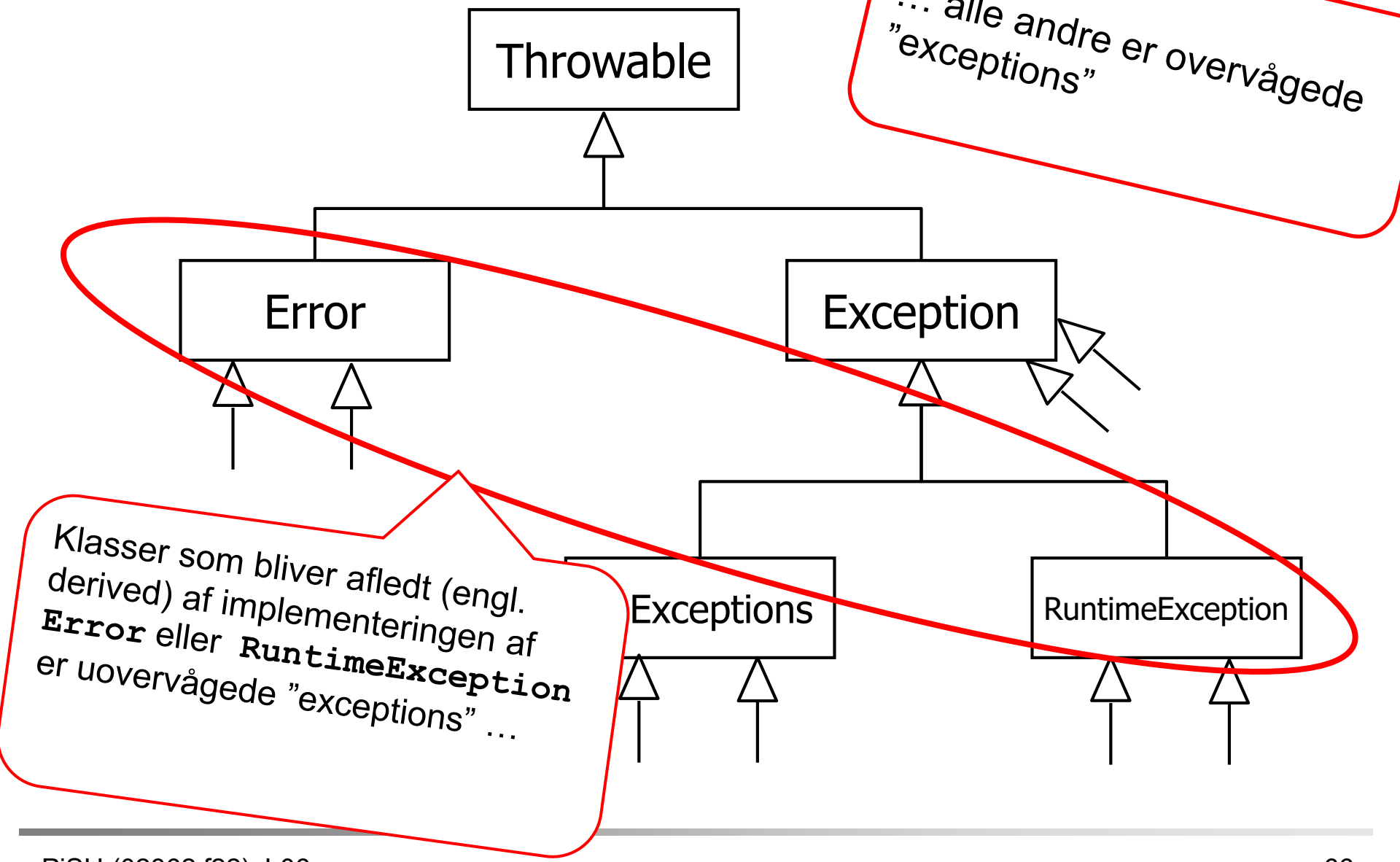
Exception: Typehierarki

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



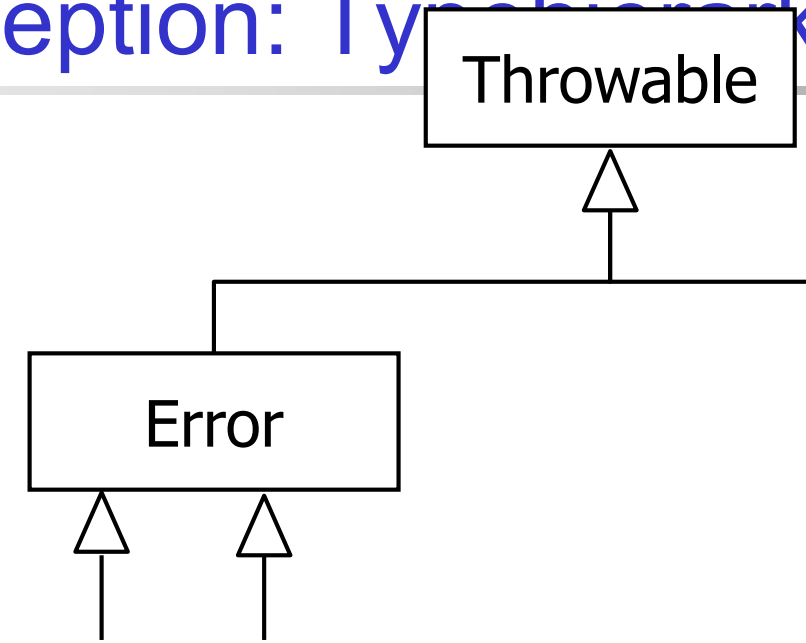
Exception: Typehierarki

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



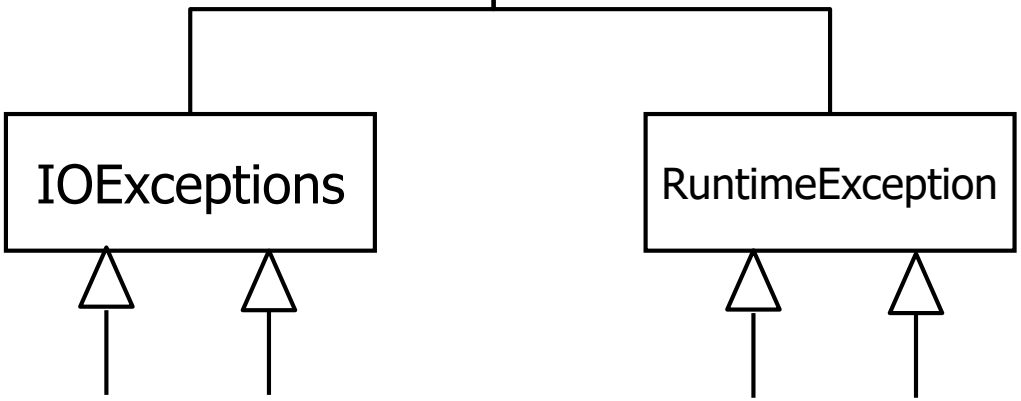
Exception: Typ-hierarki

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler



- IOError
- StackOverflowError
- OutOfMemoryError
- …

- Reflective-Operation-Exception
- SQLExeption
- …



- FileNotFoundException
- EOFException
- …

- ClassCastException
- IllegalArgumentException
- NullPointerException
- IndexOutOfBoundsException
- …

Egne Exceptions

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Man kan selv definere exceptions:
  man afleder (extends) en klasse fra Java's
  `Exception` klasse

- Man skal så overveje om man afleder af
  - Error,
  - RuntimeException,
  - Exception eller
  - IOException

Det bliver så uovervågede (unchecked) exceptions, …

… alle andre er overvågede (checked) exceptions

Kommentarer

- I nogle tilfælde giver det mening at man specificerer også uovervågede exceptions så at APIen er eksplicit om mulige exceptions (se fx. vore eksempler IllegalArgumentException)!

- Man skal **ikke** definere alle sine exceptions som RuntimeExceptions bare fordi man ikke er nødt til at specificere dem! Javas dokumentation siger:

  ”If a client can reasonably be expected to recover from an exception, make it a checked exception. If a client cannot do anything to recover from the exception, make it an unchecked exception.”

Kommentarer

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Hvis man fanger en exception med **`catch`**, kan man kaste (**`throw`**) den samme exception igen (hvis man ikke kan behandle den selv eller kun delvis). Hvis den er overvåget, så skal man selvfølgelig specificere det (lige som man kaster en exception som man har kreeret selv)

```
try {
   // do something, which might cause an
   // exception
} catch (Exception e )  {
   // do some cleanup
   throw e;
}
```

Kommentarer

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Man kan også kreere en ny exception, med den oprindelige som parameter. Det hedder *chaining of exceptions:*

```
try {
   // do something, which might cause an
   // exception
} catch (Exception e )  {
   // do some cleanup
   throw new MyOwnException("My message", e);
}
```

Positive Exceptions

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Exceptions kan også være ”positive”: fx. når en robot når sin sidste checkpoint.
- Hvis det sker midt i en længere følge af metodeopkald. Kan man nemt komme tilbage til den første opkald, med at kaste en **`GameEnded`** exception (som så evtl. skal fanges af den yderste opkald).

Men hvis I bygger jeres software op rigtigt, så er det ikke nødvendigt.

Exceptions

DTU Compute
Department of Applied Mathematics and Computer Science
Ekkart Kindler

- Mere information

Men I kan gerne kigge selv allerede nu:
https://docs.oracle.com/javase/tutorial/essential/exceptions/

- Flere exceptions i koden vedr. databasetilknytning (nedenfor)