Universität Augsburg
Institut für Informatik

Deckblatt: Übung zur Vorlesung Informatik 2 SS 2018
Fakultät für Angewandte Informatik
Lehrprofessur für Informatik
Prof. Dr. Lorenz, Marius Brendle, Johannes Metzger

Hinweis: Es sind alle Felder auszufüllen! Abgabe der Übungsblätter immer **mittwochs** (Ausnahme wenn Feiertag: donnerstags) bis **spätestens 12:00 Uhr** in die entsprechend gekennzeichneten Briefkästen der Veranstaltung im Erdgeschoss des Instituts für Informatik (Gebäude N). Zuwiderhandlung wird mit Strafe geahndet! (Punktabzug)

Übungsblatt	

(hier die Nummer des bearbeiteten **Übungsblatts eintragen**)

	Übung 01 (1057 N) Montag 08:15 - 09:45 Uhr (Isabell Rücker)
	Übung 02 (1056 N) Montag 14:00 - 15:30 Uhr (Henning Cui)
	Übung 03 (1057 N) Montag 15:45 - 17:15 Uhr (Josef Kircher)
	Übung 04 (1054 N) Montag 17:30 - 19:00 Uhr (Mosaab Slimani)
	Übung 05 (1057 N) Montag 17:30 - 19:00 Uhr (David Hacker)
	Übung 06 (1055 N) Dienstag 12:15 - 13:45 Uhr (André Schweiger)
X	Übung 07 (1054 N) Dienstag 17:30 - 19:00 Uhr (Benjamin Sertolli)
	Übung 08 (1057 N) Dienstag 17:30 - 19:00 Uhr (Dat Le Thanh)
	Übung 09 (1054 N) Mittwoch 08:15 - 09:45 Uhr (Erik Pallas)
	Übung 10 (1055 N) Mittwoch 08:15 - 09:45 Uhr (Moritz Feldmann)
	Übung 11 (1054 N) Mittwoch 10:00 - 11:30 Uhr (Denise Böhm)
	Übung 12 (1056 N) Donnerstag 08:15 - 09:45 Uhr (Florian Magg)
	Übung 13 (1054 N) Donnerstag 15:45 - 17:15 Uhr (Marvin Drexelius)
	Übung 14 (1054 N) Donnerstag 17:30 - 19:00 Uhr (Patrick Eckert)
	Übung 15 (1057 N) Donnerstag 17:30 - 19:00 Uhr (Alexander Szöke)
	Übung 16 (1057 N) Freitag 08:15 - 09:45 Uhr (Philipp Braml)
	Übung 17 (1054 N) Freitag 10:00 - 11:30 Uhr (Elisabeth Korndörfer)
	Übung 18 (1054 N) Freitag 12:15 - 13:45 Uhr (Philipp Häusele)
	Übung 19 (1056 N) Freitag 12:15 - 13:45 Uhr (Maximilian Demmler)
	Übung 20 (1054 N) Freitag 14:00 - 15:30 Uhr (Florian Straßer)

(hier die eingeteilte **Übungsgruppe ankreuzen**)

Teamnummer	6

(hier die Nummer des eingeteilten **Teams eintragen**)

Tarik Selimovic
Anton Lydike
Dominic Ceanak

(hier die **Vor- und Nachnamen** aller Teammitglieder **eintragen**)

Aufgabe		
Aufgabe		
Aufgabe		
Aufgabe		
Gesamt		(vom Tutor auszufüllen)

Übungsblatt 3

9)

A)

```
package aufgabe9;

import aufgabe9.Circle;
import aufgabe9.Point;

import java.util.zip.DataFormatException;

public class A {

    public static void main(String args[]) {

        if(args.length % 3 != 0) {
            System.out.println("Die Anzahl der Eingabeparameter ist nicht durch 3 teilbar!");
            System.exit(0);
        }

        int n = 0;
        for (int i = 0; i < args.length; i = i+3) {
            try {
                double x = Double.parseDouble(args[i]);
                double y = Double.parseDouble(args[i+1]);
                double r = Double.parseDouble(args[i+2]);

                Point p = new Point(x,y);
                Circle c = new Circle(r, p);
                n++;
            } catch (NumberFormatException e) {
                System.err.println("Couldn't convert!");
            } catch (DataFormatException e) {
                System.err.println("Could not create circle!" + e.getMessage());
            }
        }

        System.out.println("Es wurde(n) " + n + " Kreis(e) erzeugt.");
    }

}
```

B)

```
package aufgabe9;

import aufgabe9.Contact;

import java.util.zip.DataFormatException;

public class B {

    public static void main(String args[]) {
        if (args.length < 1) {
            System.err.println("Fehlende Eingabeparameter!");
            System.exit(0);
        }
        Contact contact;
        try {
            contact = new Contact(args[0]);

        } catch (DataFormatException e) {
            System.err.println("Kontakt konnte nicht erstellt werden: " + e.getMessage());
            return;
        }

        if (args.length > 1) {
            try {
                contact.setBirthday(args[1]);
            } catch (DataFormatException e) {
                System.err.println("Geburtstag konnte nicht gesetzt werden: " + e.getMessage());
            }
        }
        for (int i = 2; i < args.length; i++) {
            try {
                contact.addTelephone(args[i]);
            } catch (DataFormatException e) {
                System.err.println("Telefon konnte nicht gesetzt werden: " + e.getMessage());
            }
        }

        for (String telephone : contact.getTelephone()) {
            System.out.println(telephone);
        }

    }
}
```

Circle)

```
/*
 *
 * @Circle.java 03 24.04.2018 (Robert Lorenz)
 *
 * Copyright (c) 2017 Lehrprofessur für Informatik, Universität Augsburg
 *
 */

package aufgabe9;

import java.util.zip.DataFormatException;

/**
 * Durch seinen Radius spezifizierter Kreis
 *
 * @author lorenzro
 */
public class Circle {

  private double radius;
  private Point position;
  private static int count = 0;

  /**
   * Erzeugt einen neuen Kreis mit Radius 1.0
   *
   * @param position
   *            Position
   */
  public Circle(Point position) {
    this.radius = 1.0;
    this.setPosition(position);
    ++count;
  }

  /**
   * Erzeugt einen neuen Kreis mit übergebenem Radius
   *
   * @param position
   *            Position
   * @param radius
   *            übergebener Radius
   * @throws DataFormatException falls ungültige Daten übergeben wurden
   */
  public Circle(double radius, Point position) throws DataFormatException {
    this.setRadius(radius);
    this.setPosition(position);
    ++count;
  }

  /**
   * Gibt die Anzahl aller erzeugten Circle-Objkete zurück
   *
   * @return count Anzahl aller erzeugten Circle-Objkete
   */
  public static int getCount() {
    return count;
  }

  /**
   * Gibt den Radius zurück
   *
   * @return Radius des Kreises
   */
  public double getRadius() {
    return this.radius;
  }
```

```
  /**
   * Gibt die Position zurück
   *
   * @return Position des Kreises
   */
  public Point getPosition() {
    return this.position;
  }

  private static boolean checkRadius(double radius) {
    return (radius >= 0);
  }

  /**
   * Setzt den Radius, falls ein gültiger Wert übergeben wird. Gültig sind
   * positive Zahlen.
   *
   * @param radius
   *            Radius
   * @throws DataFormatException falls ungültige Daten übergeben wurden
   */
  public void setRadius(double radius) throws DataFormatException {
    if (!checkRadius(radius))
      throw new DataFormatException("Objekt Circle: Radius ungueltig");
    this.radius = radius;
  }

  /**
   * Setzt die Position.
   *
   * @param position
   *            Position
   */
  public void setPosition(Point position) {
    this.position = position;
  }

  /**
   * Berechnet die Fläche des Kreises
   *
   * @return Kreisflaeche
   */
  public double getArea() {
    return Math.PI * this.radius * this.radius;
  }

}
```

Contact)

```
package aufgabe9;
/*
 *
 * @Contact.java 03 24.04.2018 (Robert Lorenz)
 *
 * Copyright (c) 2017 Lehrprofessur für Informatik, Universität Augsburg
 *
 */

import java.util.ArrayList; /*Klasse ArrayList importieren*/
import java.util.zip.DataFormatException;

/**
 * Kontakte mit Namen (erforderlich), Geburtstag (optional) und Telefonliste
 * (optional)
 *
 * @author lorenzro
 */
public class Contact {

  private String birthday;
  private String name;
  private ArrayList<String> telephone;

  /**
   * Erzeugt einen neuen Kontakt mit dem übergebenen Namen
   *
   * @param name
   *            der Name des erzeugten Kontakts
   *
   * @throws DataFormatException falls ungültige Daten übergeben wurden
   */
  public Contact(String name) throws DataFormatException {
    setName(name);
    birthday = null; /* Standardwert setzen */
    telephone = new ArrayList<String>(); /* Leere Liste erzeugen */
  }

  /**
   * Gibt den Geburtstag zurück
   *
   * @return Geburtstag des Kontakts
   */
  public String getBirthday() {
    return this.birthday;
  }

  private static boolean checkBirthday(String birthday) {
      /* Ueberpruefung mit regulaeren Ausdruecken */
      return birthday.matches("\\d{2}\\.\\d{2}\\.\\d{4}");
  }

  /**
   * Setzt den Geburtstag neu, falls ein gültiger Wert in der Form tt.mm.yyyy
   * # mit tt=Tag zweistellig, mm=Monat zweistellig und yyyy=Jahr vierstellig
   * übergeben wird
   *
   * @param birthday
   *            der neue Geburtstag
   *
   * @throws DataFormatException falls ungültige Daten übergeben wurden
   */
  public void setBirthday(String birthday) throws DataFormatException {
    if (birthday != null && !checkBirthday(birthday))
      throw new DataFormatException("Objekt Contact: Geburtstag ungueltig");
    this.birthday = birthday;
  }
```

```java
/**
 * Gibt den Namen zurück
 *
 * @return Name des Kontakts
 */
public String getName() {
  return name;
}

private static boolean checkName(String name) {
  return (name != null && name.length() > 0);
}

/**
 * Setzt den Namen neu, falls ein gültiger Wert mit mindestens Länge 1
 * übergeben wird
 *
 * @param name
 *            der neue Name
 *
 * @throws IllegalArgumentException falls ungültige Daten übergeben wurden
 */
public void setName(String name) throws DataFormatException {
  if (!checkName(name))
    throw new DataFormatException("Objekt Contact: Name ungueltig");
  this.name = name;
}

/**
 * Gibt die Telefonliste zurück
 *
 * @return Name des Kontakts
 */
public ArrayList<String> getTelephone() {
  /* Kopie erzeugen und zurückgeben */
  ArrayList<String> copy = new ArrayList<String>();
  for (String number : this.telephone) {/* Telefonliste durchlaufen */
    copy.add(number);
  }
  return copy;
}

/* Alternative zur Rückgabe der Liste aller Telefonnummern*/
/**
 * Gibt die p-te Nummer aus der Telefonliste zurück
 *
 * @param p
 *            Index der geforderten Nummer aus der Telefonliste
 * @return p-te Nummer der Telefonlsite, falls Wert vorhanden
 *
 * @throws IndexOutOfBoundsException falls p-ter Wert nicht vorhanden
 */
public String getTelephone(int p) throws IndexOutOfBoundsException {
    return this.telephone.get(p);
}

/* Alternative zur Rückgabe der Liste aller Telefonnummern*/
/**
 * Gibt die Anzahl aller Nummern aus der Telefonliste zurück
 *
 * @return Anzahl aller Nummern aus der Telefonliste
 *
 */
public int getCountTelephone() {
    return this.telephone.size();
}

private static boolean checkTelephone(String telephone) {
  return true;
}
```

```
  /**
   * Fügt eine neue Telefonnummer hinzu
   *
   * @throws DataFormatException falls ungültige Daten übergeben wurden
   */
  public void addTelephone(String telephone) throws DataFormatException {
    if (!checkTelephone(telephone))
      throw new DataFormatException("Objekt Contact: Telefon ungueltig");
    if (this.telephone.contains(telephone)) {
      throw new DataFormatException("Objekt Contact: Telefon schon vorhanden");
    }
    this.telephone.add(telephone);
  }

  /**
   * Löscht eine Telefonnummer
   *
   */
  public void deleteTelephone(String telephone) {
    this.telephone.remove(telephone);
  }
}
```

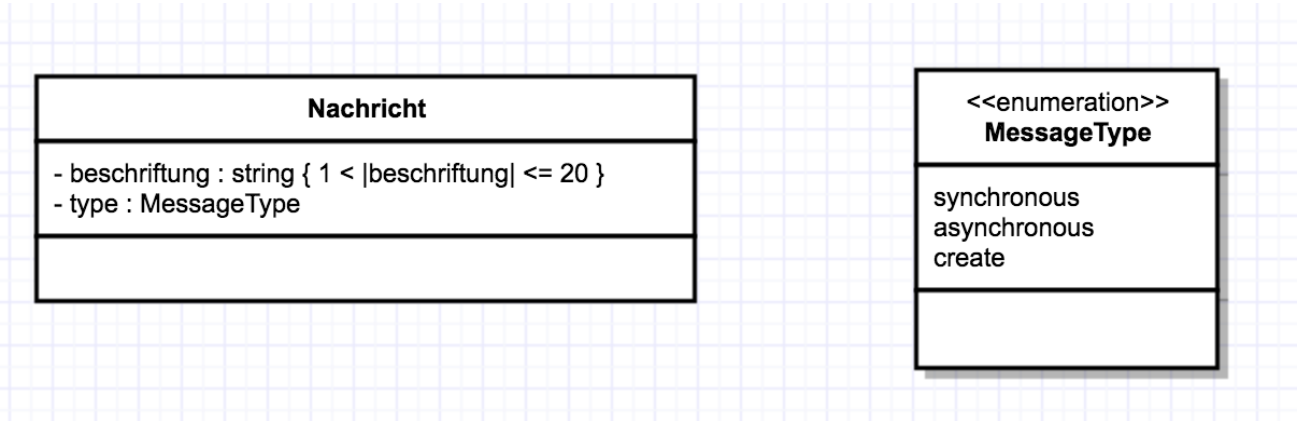
Point)

```
/*
 *
 * @Point.java 02 24.04.2018 (Robert Lorenz)
 *
 * Copyright (c) 2017 Lehrprofessur für Informatik, Universität Augsburg
 *
 */

package aufgabe9;

/**
 * Punkt mit jeweils reelwertiger x- und y-Koordinate
 *
 * @author lorenzro
 *
 */
public class Point {

  private double x;
  private double y;

  /**
   * Erzeugt einen Punkt mit den übergebenen x- und y-Koordinaten.
   *
   * @param x der Wert der x-Koordinate des erzeugten Punktes
   * @param y der Wert der y-Koordinate des erzeugten Punktes
   *
   */
  public Point(double x, double y)  {
      setX(x);
      setY(y);
  }


  /**
   * Setzt Wert der x-Koordinate
   *
   * @param x der neue Wert der x-Koordinate
   *
   */
  public void setX(double x) {
    this.x = x;
  }

  /**
   * Setzt Wert der y-Koordinate
   *
   * @param y der neue Wert der y-Koordinate
   *
   */
  public void setY(double y) {
    this.y = y;
  }


  /**
   * Gibt Wert der x-Koordinate zurück
   *
   * @return Wert der x-Koordinate
   */
  public double getX() {
    return this.x;
  }

  /**
   * Gibt Wert der y-Koordinate zurück
   *
   * @return Wert der y-Koordinate
```

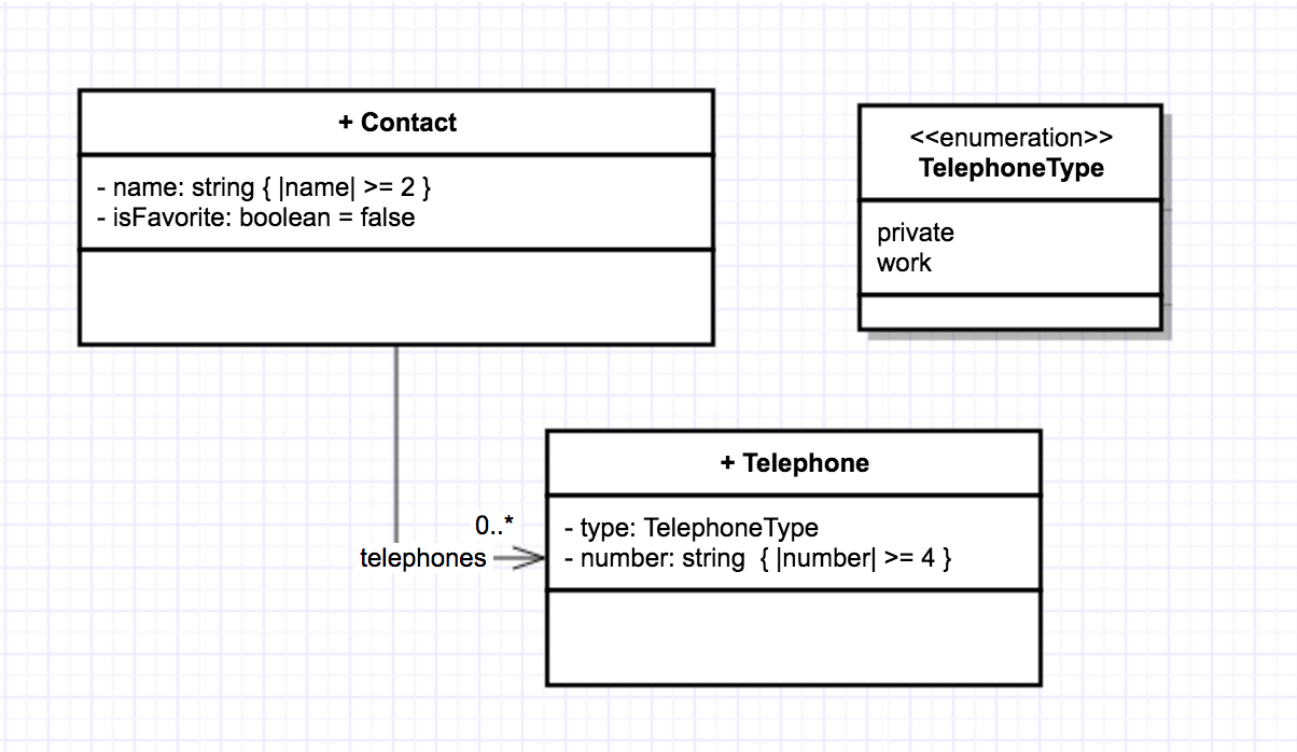
```
   */
  public double getY() {
    return this.y;
  }

}
```

10)

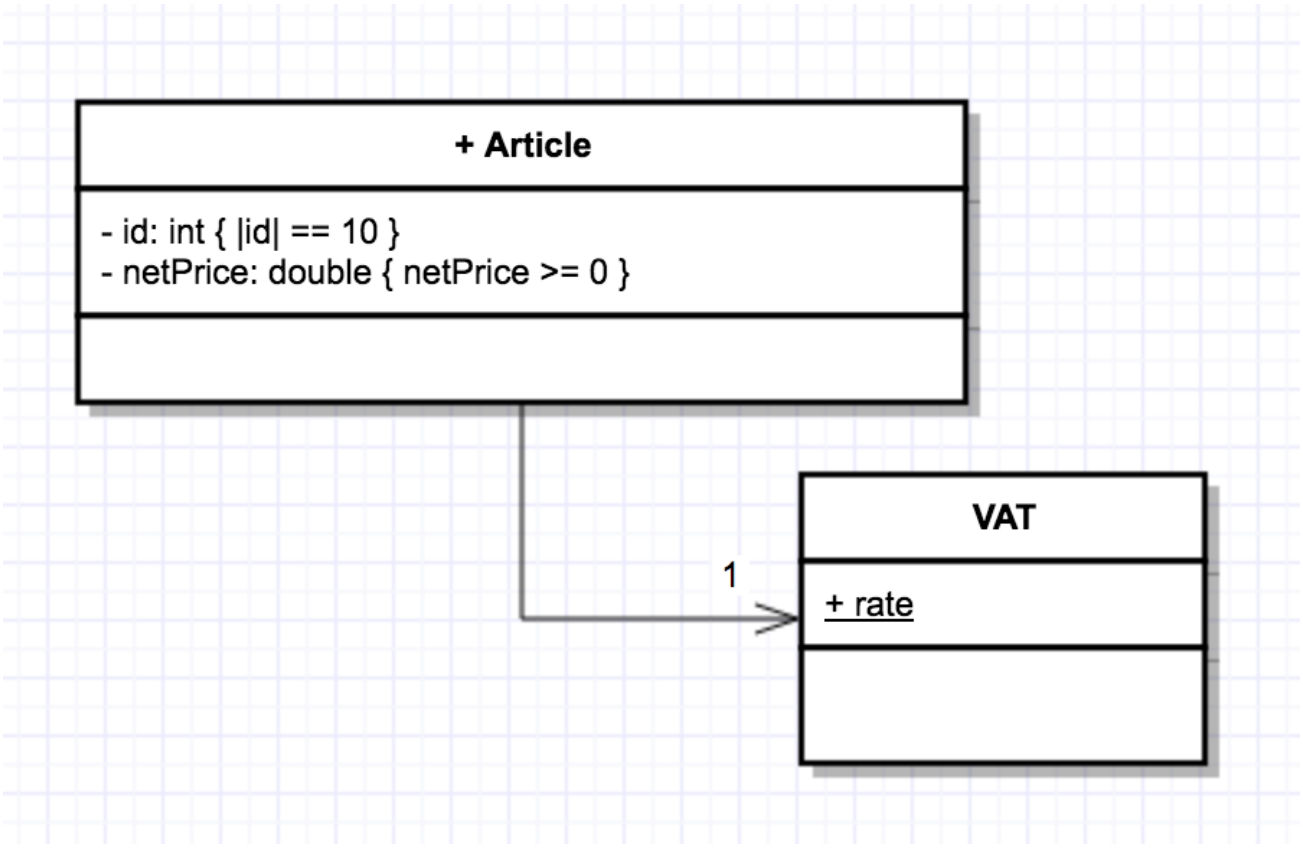
a)



b)

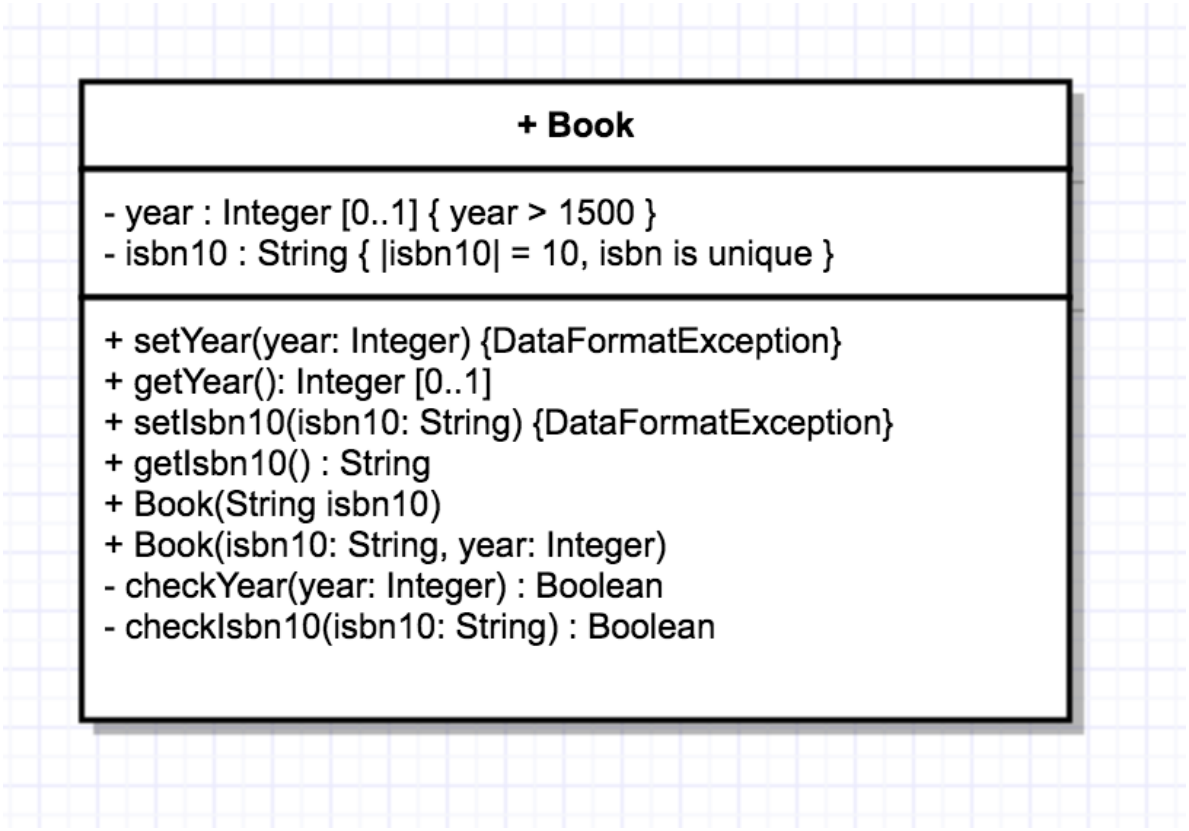


c)

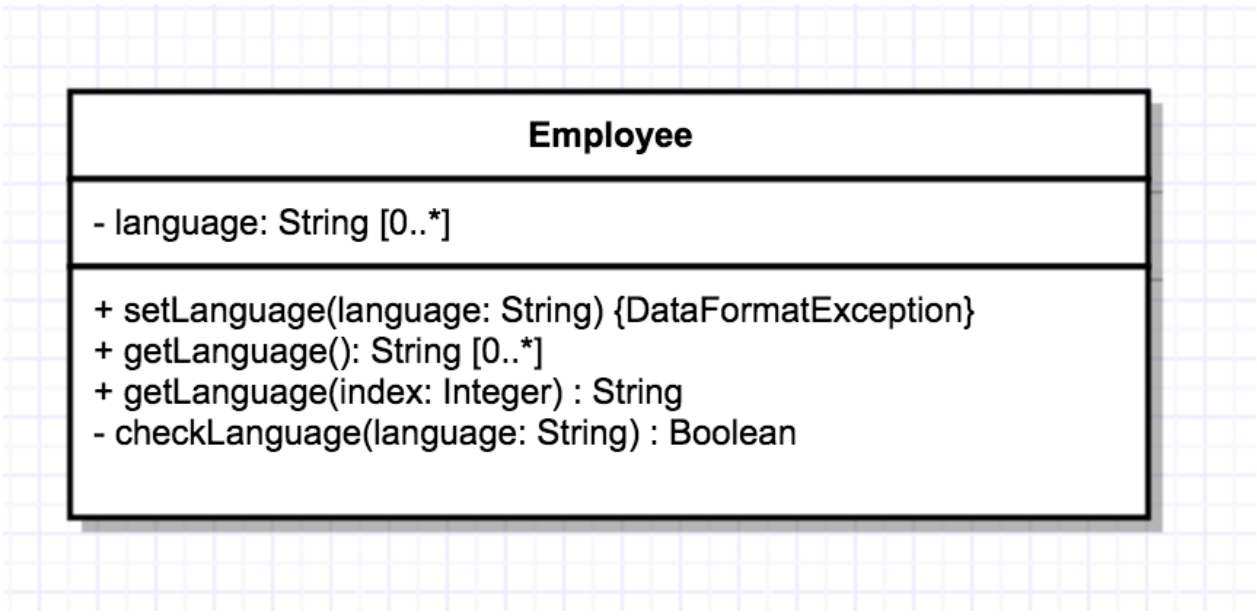

+ Article
- id: int { |id| == 10 }
- netPrice: double { netPrice >= 0 }
VAT
1
+ rate

11)

a)


+ Book
- year : Integer [0..1] { year > 1500 }
- isbn10 : String { |isbn10| = 10, isbn is unique }
+ setYear(year: Integer) {DataFormatException}
+ getYear(): Integer [0..1]
+ setIsbn10(isbn10: String) {DataFormatException}
+ getIsbn10() : String
+ Book(String isbn10)
+ Book(isbn10: String, year: Integer)
- checkYear(year: Integer) : Boolean
- checkIsbn10(isbn10: String) : Boolean

b)


Employee
- language: String [0..*]
+ setLanguage(language: String) {DataFormatException}
+ getLanguage(): String [0..*]
+ getLanguage(index: Integer) : String
- checkLanguage(language: String) : Boolean

12)

Hobby)

```
package aufgabe12;

import java.util.zip.DataFormatException;

public class Hobby {


    private String name;
    private int priority = 1;

    public Hobby(String name) throws DataFormatException {
        this.setName(name);
    }

    public Hobby(String name, int priority) throws DataFormatException {
        this.setName(name);
        this.setPriority(priority);
    }

    public String getName() {
        return name;
    }

    public int getPriority() {
        return priority;
    }

    private void setName(String name) throws DataFormatException {
        if(checkName(name)) {
            this.name = name;
        } else {
            throw new DataFormatException("Name ist ungueltig!");
        }
    }

    public void setPriority(int priority) throws DataFormatException{
        if(checkPriority(priority)) {
            this.priority = priority;
        } else {
            throw new DataFormatException("Priority ist ungueltig!");
        }
    }

    private static boolean checkName(String name) {
        return !name.isEmpty();
    }

    private static boolean checkPriority(int priority) {
        return priority > 0 && priority < 6;
    }
}
```

Student)

```
package aufgabe12;

import java.util.ArrayList;
import java.util.List;
import java.util.zip.DataFormatException;

public class Student {
    private List<String> email = new ArrayList<>();

    public void addEmail(String email) throws DataFormatException {
        if(!checkEmail(email) ) {
            throw new DataFormatException("Email ist ungueltig. email=" + email);
        } else {
            if (!this.email.contains(email)) {
                this.email.add(email);
            } else {
                throw new DataFormatException("Email bereits vorhanden. email=" +email);
            }
        }
    }

    public List<String> getEmail() {
        return this.email;
    }

    public void deleteEmail(String email) {
        this.email.remove(email);
    }

    public int getEmailCount() {
        return this.email.size();
    }

    private static boolean checkEmail(String email) {
        return email.indexOf("@") >= 1 && email.length() > (email.indexOf("@") + 1);
    }
}
```