Workout

Erstellen Sie eine einfache Fitness-App.

Einleitung

In dieser App werden alle Fitness-Aktivitäten durch `BasicWorkout` oder davon abgeleitete Klassen dargestellt.

Ein `BasicWorkout` ist eine unspezifizierte/allgemeine Fitness-Aktivität mit

- Datum
- Dauer (s)
- Intensität (0-10)
- Verbrannter Energie (kCal)
- Beschreibung (max. 100 Zeichen)

Davon abgeleitet gibt es die 2 speziellen Fitness-Aktivitäten `Running` und `PushUp`.

`Running` hat als zusätzliche Eigenschaft die gelaufene Strecke `distance`. Außerdem werden Energie und Intensität anhand von gelaufener Strecke und Dauer errechnet.

`PushUp` hat als zusätzliche Eigenschaft die Anzahl der Wiederholungen. Sowohl Energie als auch Intensität werden anhand der Wiederholungen errechnet.

Die Klasse `Main` simuliert die App indem sie einige Fitness-Aktivitäten erstellt und die Gesamt-Kalorien sowie die mittlere Intensität errechnet.

Aufgabe

Implementieren Sie die Klassen `BasicWorkout`, `Running` und `Main` wie in den javadocs spezifiziert. Siehe auch Abb. 1 für einen Überblick.

Beispiele

(Text in rot = Benutzereingabe)

Testausgabe von `Main.main()`

```
01.04.2018: [*******   ] "Bike trip to Krems" 2.3h, 1200kcal
08.05.2018: [*****     ] "Digging a hole" 0.3h, 500kcal
07.05.2018: [*******   ] "Evening run from work to home" 1.0h, 700kcal 12.0km
---
total energy: 2400kCal
---
mean intensity: 6.3
```

Hint

- Die Testfälle testen lediglich die Gesamtfunktionalität von `Main`.
- Sie können folgenden Code für `BasicWorkout`verwenden

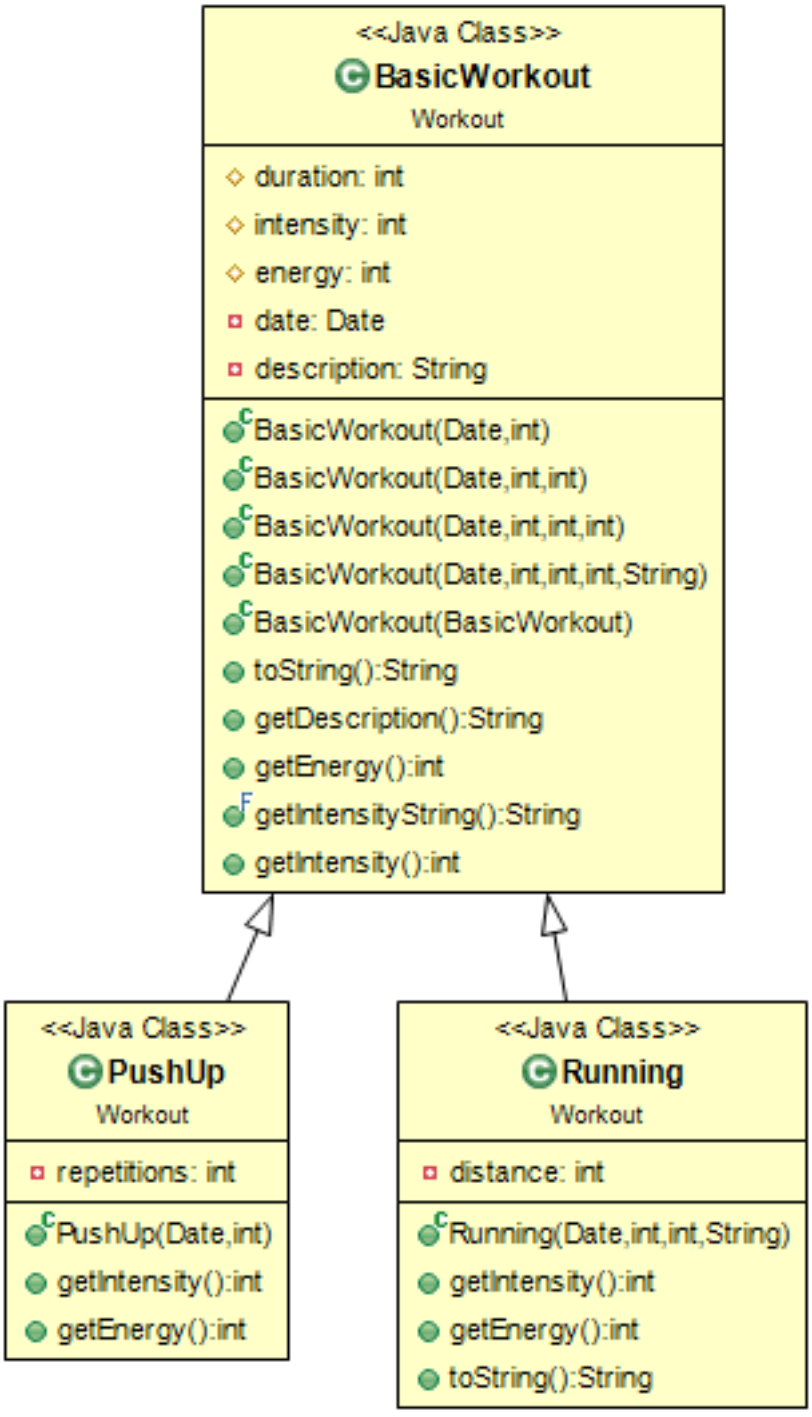

Figure 1: Klassendiagrmm `Workout`

```
@Override
public String toString() {
    return String.format("%s: %s \"%s\" %.1fh, %dkcal ",
        date, getIntensityString(), getDescription(), duration/3600., getEnergy());
}
```

```
public BasicWorkout(Date date, int duration, int intensity, int energy, String description) {
    this(date, duration, intensity, energy);
    this.description = (description == null) ? null : String.format("%.100s", description);

}
```

- Sie können folgenden Code für `Main.getDemoData()` verwenden

```
public static BasicWorkout[] getDemoData(){

    return new BasicWorkout[] {
            new BasicWorkout(new Date(2018, 04, 1), 8200, 7, 1200, "Bike trip to Krems"),
            new BasicWorkout(new Date(2018, 05, 8), 1200, 5, 500, "Digging a hole"),
            new Running(new Date(2018, 05, 7), 3600, 12000, "Evening run from work to home"),

    };
}
```

Documentation - BasicWorkout

`protected int duration`

the duration in seconds of this workout. Must be >0.

`protected int intensity`

intensity (out of the range 1-10) of this workout.

`protected int energy`

the energy burned in this workout in kcal. Must be >0.

`private Date date`

the date of the (start of) this workout.

`private String description`

a max. 100 character description of this workout.

`public BasicWorkout(Date date, int duration)`

Constructs a basic workout on a specified date and of a specified duration.

- **Parameters:**
  - `date` — the date on which this workout started
  - `duration` — the duration of this workout in s

`public BasicWorkout(Date date, int duration, int intensity)`

Constructs a basic workout on a specified date, duration and intensity.

- **Parameters:**
  - `date` — the date on which this workout started
  - `duration` — the duration of this workout in s
  - `intensity` — the intensity of this workout

`public BasicWorkout(Date date, int duration, int intensity, int energy)`

Constructs a basic workout on a specified date, duration, intensity and energy.

- **Parameters:**
  - `date` — the date on which this workout started
  - `duration` — the duration of this workout in s
  - `intensity` — the intensity of this workout
  - `energy` — the energy burned during this workout in kcal

public BasicWorkout(Date date, int duration, int intensity, int energy, String description)

Constructs a basic workout on a specified date and of a specified duration.

- **Parameters:**
  - `date` — the date on which this workout started
  - `duration` — the duration of this workout in s
  - `intensity` — the intensity of this workout
  - `energy` — the energy burned during this workout in kcal
  - `description` — the description of this workout

public BasicWorkout(BasicWorkout bw)

Creates a deep copy of a basic workout.

- **Parameters:** `bw` — the original workout to be copied

@Override public String toString()

Creates a String representation of this workout.

public String getDescription()

getter method for the description.

if no description is available `<no description>` is returned.

- **Returns:** the description of this workout or
  <no description> if no description is available

public int getEnergy()

getter method for the energy burned during this workout.

- **Returns:** the energy in kcal of this workout

public final String getIntensityString()

returns a String representation of the intensity of this workout.

The intensity is represented with stars enclosed in brackets from `[*       &n` for intensity 1 to `[**********]` for intensity 10.

- **Returns:** the String representation of the intensity of this workout.

public int getIntensity()

getter method for the intensity of this workout.

- **Returns:** the intensity.

Documentation - Running

`public Running(Date date, int duration, int distance, String description)`

creates a run on specified date with a specified duration, distance and description.

- **Parameters:**
  - `date` — the date of this run
  - `duration` — the duration of this run in s
  - `distance` — the distance of this run in m
  - `description` — the description of this run

`private int distance`

distance of this running workout in m.

`@Override public int getIntensity()`

getter method for the intensity of this run.

The intensity is based on the pace (min/km) according to the table:

intensity levels based on pace

pace [min/km]

intensity

10

9

8

7

6

5

4

3

2

other

1

`@Override public int getEnergy()`

estimates the energy burned during this run.

the energy is estimated as the intensity * 100 * duration in minutes.

`@Override public String toString()`

returns a String representation of this run.

Appends the distance to the string representation of a `BasicWorkout`

Documentation Main

`private static double meanIntensity(BasicWorkout[] data)`

calculates the mean intensity of all workouts in `data`

- **Parameters:** `data` — the workouts
- **Returns:** the mean intensity

`public static int totalEnergy(BasicWorkout[] data)`

calculates the total energy of all workouts in `data`

- **Parameters:** `data` — workouts
- **Returns:** the total energy

`public static void print(BasicWorkout[] data)`

prints all workouts in data, one per line

- **Parameters:** `data` — the workouts to print

`public static BasicWorkout[] getDemoData()`

create some demo data

- **Returns:** the demo data