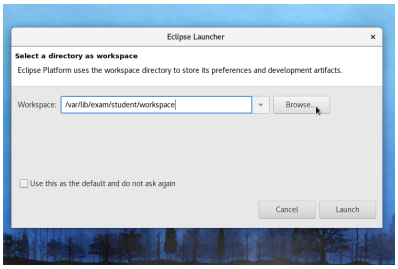
Anmelden und Eclipse starten

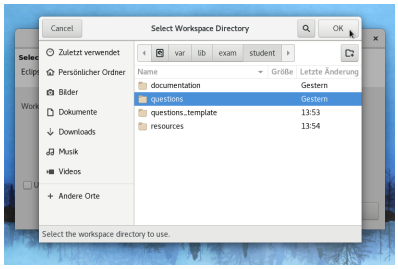
1. Sobald die Prüfung startet, können Sie sich an Ihrem Computer anmelden. Geben Sie zuerst Ihren vollen Namen und im nächsten Schritt Ihren NETHZ-Namen ein.

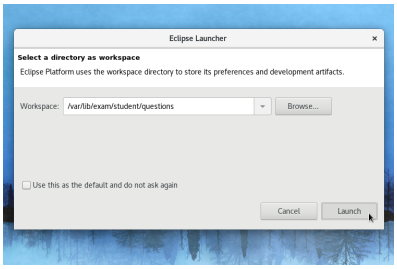
2. Starten Sie Eclipse, indem Sie oben links auf "Aktivitäten" klicken und dann im Suchfeld "Eclipse" eingeben. Wählen Sie "Eclipse" (*nicht* "Eclipse C/C++"). Warten Sie, bis Eclipse gestartet ist. Dies kann einige Minuten in Anspruch nehmen, also nutzen Sie die Zeit, um sich einen Überblick über die Aufgaben zu verschaffen.

3. Wenn sich das Fenster "Eclipse Launcher" öffnet, wählen Sie den Prüfungs-Workspace folgendermassen aus:

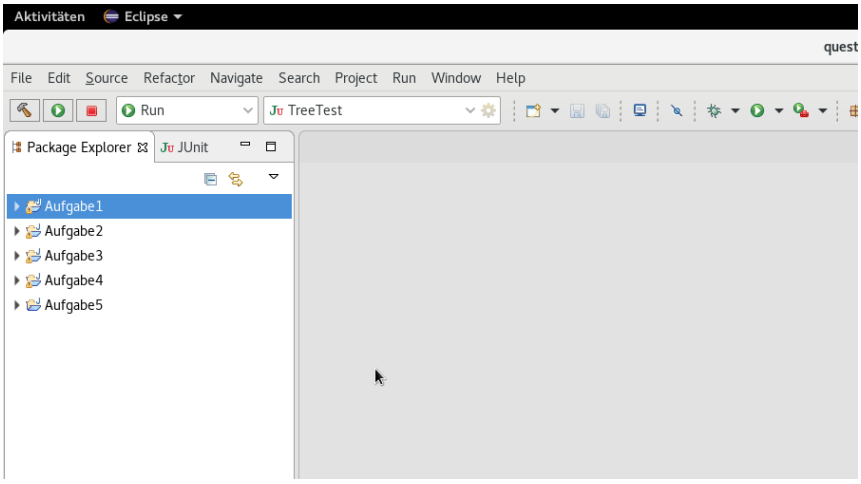
   Klicken Sie auf "Browse..." und wählen Sie dann im Auswahldialog den "questions"-Ordner aus. Klicken Sie oben rechts auf "OK". Im Fenster "Eclipse Launcher" sollte unter "Workspace" jetzt folgender Pfad stehen: "/var/lib/exam/student/questions". Klicken Sie "Launch".







4. Wenn Eclipse fertig gestartet ist, sehen Sie den Willkommens-Bildschirm. Klicken Sie oben rechts auf "Workbench". Nun sollten Sie links die fünf Projekte "Aufgabe 1" bis "Aufgabe 5" sehen.



Aufgabe 1

Gegeben seinen zwei Strings `s` und `t`, die aus S und T unterschiedlichen Buchstaben bestehen. (Das heisst, dass `s` und `t` keine Buchstaben gemeinsam haben und kein Buchstabe doppelt oder häufiger auftritt.) Schreiben Sie ein Programm, das alle $(S+T)!/(S! \times T!)$ Verzahnungen (Interleavings) generiert.

Beispiel: `s` = `"12"`, `t` = `"ab"`. Dann lauten die 24/4 = 6 Verzahnungen: `12ab`, `1a2b`, `1ab2`, `a12b`, `a1b2`, und `ab12`.

Vervollständigen Sie die Methode `verzahnungen()` in der Klasse `Verzahnungen` im Projekt "Aufgabe1", so dass sie die Verzahnungen für die beiden String-Parameter `s` und `t` generiert und in einem `java.util.Set<String>` zurückgibt. Sie können annehmen, dass weder `s` noch `t` der leere String sind (also $S, T > 0$ gilt) und $S, T < 128$ gilt.

Aufgabe 2

Schreiben Sie ein Programm, welches die Primfaktorzerlegung einer ganzen Zahl `n` berechnet. Die Primfaktorzerlegung ist die Darstellung von `n` als Produkt aus Primzahlen. Diese heissen *Primfaktoren* von `n`. Um die Zerlegung für positive *und* negative Zahlen darstellen zu können, soll Ihr Programm immer 1 oder -1 zu den Primfaktoren hinzufügen.

Beispiele:

- `n` = 36. Die Primfaktoren sind 1, 2, 2, 3, 3.
- `n` = -13. Die Primfaktoren sind -1, 13.

Vervollständigen Sie die Methode `Primfaktoren.primfaktoren()`, so dass sie den Parameter `n` in seine Primfaktoren zerlegt und diese in einer `java.util.List<Integer>` zurückgibt. Die Primfaktoren sollen von klein nach gross sortiert in der Liste stehen.

Beachten Sie: Die erste Zahl in der Liste soll immer 1 sein, falls `n` positiv ist, und -1, falls `n` negativ ist. Danach folgen die "richtigen" Primfaktoren. Falls `n` = 0, soll die Methode eine leere Liste zurückgeben.

Aufgabe 3

Durch eine Indiskretion sind Ihnen vertrauliche Daten des Bienenzüchterverbandes zugespielt worden. In einer Datei finden Sie Informationen über die Mitglieder des Leitungsausschusses des Verbandes. Für jedes Mitglied gibt es eine Zeile in der Sie finden:

- Den Nachnamen des Mitglieds (ein Wort). Sie können davon ausgehen, dass alle Mitglieder unterschiedlich heissen.
- Das Land, das dieses Mitglied vertritt. Es wird die für Internet-Domainnamen übliche Abkürzung verwendet, also 2 Buchstaben wie "fr" für Frankreich und "uk" für das Vereinigte Königreich.
- Die offiziell gezahlten Sitzungsgelder (in EUR), als positive ganze Zahl.
- Die nicht versteuerten "Sonderzahlungen" (in EUR), als positive ganze Zahl.

Schreiben Sie ein Programm, welches aus diesen Daten folgende Informationen berechnet:

1. Den Namen des Mitglieds, das die höchste Gesamtzahlung (Sitzungsgeld + Sonderzahlung) erhalten hat, und den Betrag.
2. Den Namen des Mitglieds, für welches die Sonderzahlung den grössten Anteil an der Gesamtzahlung ausmachte, und den Prozentsatz.
3. Das Land, dessen Mitglieder die höchste Summe an Sonderzahlungen erhielten, sowie den Betrag.

Falls mehrere Mitglieder oder Länder für einen der Punkte in Frage kommen, kann das Programm einen beliebigen von ihnen ausgeben.

Vervollständigen Sie die `Bienen.analyze()`-Methode. Die Methode hat zwei `File`-Parameter. Der erste bezeichnet die Eingabe-Datei, aus der Sie die Daten lesen sollen, und der zweite die Ausgabe-Datei, in welche Sie die oben genannten Informationen schreiben sollen. Ihr Programm muss nur korrekt formatierte Eingabe-Dateien unterstützen. Ein Beispiel einer solchen Datei finden Sie im Projekt unter dem Namen "bienen.txt". Exceptions im Zusammenhang mit Ein- und Ausgabe können Sie ignorieren.

Jede der Informationen 1.–3. soll (in dieser Reihenfolge) auf einer separaten Zeile ausgegeben werden. Alle Zahlen sollen arithmetisch gerundet als ganze Zahlen ausgegeben werden. Trennen Sie Name (bzw. Land) und Zahl jeweils durch ein einzelnes Leerzeichen.

Aufgabe 4

In dieser Aufgabe erweitern Sie eine vorgegebene Implementierung einer Baum-Datenstruktur für `int`-Werte um zwei Methoden.

Die Klasse `Tree` repräsentiert einen ternären Baum, das heisst, jeder Knoten im Baum hat bis zu 3 Kindknoten. Es gibt zwei Arten von Knoten, welche als zwei verschiedene Klassen abgebildet sind. Beide Klassen implementieren das Interface `Node`:

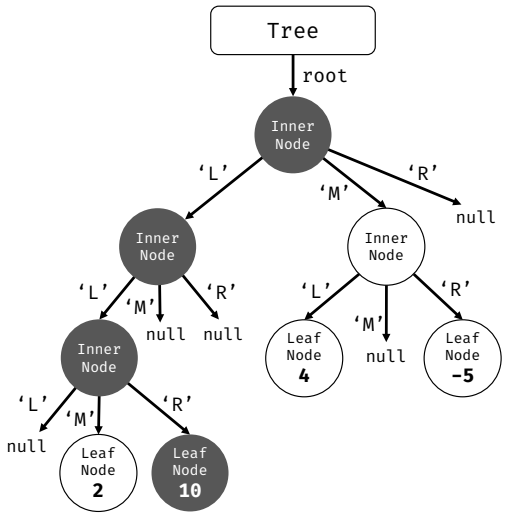
- Ein `LeafNode` repräsentiert ein Blatt im Baum. Diese Knoten speichern einen `int`-Wert und besitzen keine Kindknoten.
- Ein `InnerNode` repräsentiert einen Knoten im Innern des Baums, welcher bis zu drei Kindknoten hat, aber keinen eigenen `int`-Wert speichert. Die (bis zu) drei Kindknoten werden von links nach rechts mit `'L'`, `'M'` und `'R'` bezeichnet.

Die `Tree`-Klasse enthält eine Methode `insert(int value, String path)`, welche den Wert `value` in den Baum einfügt. Die Stelle, an welcher der Wert eingefügt wird, ist durch einen "Pfad"-String gegeben, welcher aus einer Folge von `'L'`, `'M'` und `'R'` besteht (andernfalls ist der String kein gültiger Pfad). Der Pfad `"LLR"` bedeutet zum Beispiel, dass vom Wurzelknoten aus zuerst der linke Knoten, von diesem Knoten aus nochmals der linke und schliesslich der rechte Knoten gewählt wird um den Wert einzufügen. Im Bild unten sind diese vier Knoten dunkel dargestellt. Alle Knoten, welche noch nicht existieren, werden von `insert()` erstellt.

Die folgende Serie von Aufrufen erstellt den Baum, der rechts abgebildet ist:

```
Tree tree = new Tree(0);
tree.insert( 2, "LLM");
tree.insert(10, "LLR");
tree.insert(99, "M");
tree.insert( 4, "ML");
tree.insert(-5, "MR");
```

Beachten Sie, dass der Wert 99 an der Stelle `"M"` von den späteren Aufrufen überschrieben wird, da ein innerer Knoten keinen Wert hat. Das selbe gilt für den Wert 0, mit dem der Baum initialisiert wird.



a) Implementieren Sie die `Tree.lookup()`-Methode, welche für einen gegebenen Pfad den Wert an dieser Stelle zurückgibt. Ein leerer Pfad bezeichnet den Wurzelknoten des Baums. Der Methodenkommentar für `lookup()` beschreibt die Ausnahmefälle.

   Beispiel: Für den Baum oben soll `tree.lookup("ML")` den Wert 4 zurückgeben.

b) Implementieren Sie die `Tree.toList()`-Methode. Diese gibt alle Werte im Baum (von links nach rechts) in einer Liste zurück.

   Beispiel: Für den Baum oben soll `tree.toList()` die Liste [2, 10, 4, −5] zurückgeben.

Tipp: Sie dürfen alle Klassen, und auch das `Node`-Interface, um weitere Methoden ergänzen.

Aufgabe 5

Diese Aufgabe ist keine Programmieraufgabe, aber Sie sollen Ihre Lösungen in den Dateien "EbnfTeilA.java" und "EbnfTeilB.java" im Projekt "Aufgabe 5" abgeben. (Das macht die Korrektur einfacher.) Lesen Sie den Klassenkommentar von `EbnfTeilA` *durch und halten Sie sich genau an die Anweisungen um Punkte zu erhalten.*

Entscheiden Sie für die EBNF-Beschreibungen in a) und b), welche der gegebenen Wörter legal sind. Die Liste der Wörter befinden sich jeweils in den oben genannten Dateien.

a) **EBNF-Beschreibung** *roemisch*:

is ⇐ [`I`] [`II`]
acht ⇐ `V` [*is*] | `IV` | *is*
roemisch ⇐ { `X` } (`IX` | *acht*)

b) **EBNF-Beschreibung** *rekursiv*:

rekursiv ⇐ `UN` *rekursiv* `ER` | [`RE`] *rekursiv* | `KURSIV`

Wir wünschen Ihnen alles Gute für den Rest der Prüfungssession und das nächste Semester!

— Ihr "Einführung in das Programmieren"-Team.