Übungsblatt 7

Aufgabe 1

1. Bei primitiven Variablen wird der ihr zugewiesene Wert direkt in dem für sie reservierten Speicherbereich gespeichert.

Referenzvariablen hingegen speichern eine Referenz, die auf ein bereits erzeugtes Objekt im Heap verweist.

Einige Datenstrukturen in Bibliotheken können ausschließlich mit Objekten und dementsprechend nicht mit primitiven Werten arbeiten. Um einen primitiven Wert in ein nutzbares Objekt „zu umhüllen“ werden Wrapper-Klassen verwendet.

1. 1 paule

Datentyp: Hamster, (Territory, Location, Direction, Integer)

Werte: territoryPaule = game.getTerritory; locationPaule = newLocation(1,1); directionPaule = Direction.EAST; grainCountPaule = 0;

2 Long.valueOf(24756)

Datentyp: Long

Wert: 24756

3 !paule.mouthEmpty()

Datentyp: boolean

Wert: false

4 !paule.frontIsClear() || paule.grainAvailable()

Datentyp: beide boolean

Werte: !paule.frontIsClear ist false; paule.grainAvailable() ist true, also ist der Ausdruck true

5 5-3.5f

Datentyp: 5 ist Integer, 3.5f ist float. Das Ergebnis ist ebenfalls float

Werte: Ergebnis: 1.5f

6 false != !false

Datentyp: boolean

Wert: true

7 „The size of the territory is“ + 3 + „x“ + 3 + „!“

Datentyp: String

Wert: The size of the territory is 3x3!

8 paule.getLocation()

Datentyp: Location

Wert: location(1,1)

9 paule.getLocation().getColumn() + paule.getLocation().getRow()

Datentyp: beide Integer, Ergebniss auch Integer

Werte: paule.getLocation().getColumn() = 1; paule.getLocation().getRow() = 1

Ergebnis: 2

1. getTerritory() und getColumnCount() haben zwei unterschiedliche Rückgabetypen (Location und Integer) weshalb man sie nicht unmittelbar aufeinander aufrufen und sofort als integer speichern kann, da es dabei zu einem Kompilierungsfehler kommt. Um die Anzahl von Columns zu speichern könnte man…

Aufgabe 2

1. Die Klasse ist semantisch noch nicht korrekt, da „numberOfInhabtitans“ zwar deklariert, allerdings noch nicht im Konstruktor (wegen final) initialisiert wurde, wodurch es bei der Abfrage „getNumberOfInhabitants()“ zu einem semantischen Fehler kommt. Durch die Nicht-Initialisierung sind die Objekte der Klasse ebenfalls noch nicht unveränderlich. Eine deep copy ist allerdings nicht nötig um die Objekte unveränderlich zu machen, da integer von Java schon immutable sind.

Verbesserung:

public final class House {

private final int numberOfInhabitants;

private final int numberOfBedrooms;

public House(final int numberOfInhabitants, final int numberOfBedrooms) {

this.numberOfInhabitants = numberOfInhabitants;

this.numberOfBedrooms = numberOfBedrooms;

}

public int getNumberOfInhabitants() {

return numberOfInhabitants;

}

public int getNumberOfBedrooms() {

return numberOfBedrooms;

}

}

1. Zeile 63 und 79 Setter: über Setter in einer Klasse kann man, wie der Name schon sagt, über den Operationsaufruf, Name sowie Geburtsdatum einfach verändern.

Zeile 10 „final“ fehlt. Es könnte sein, dass die Werte über eine subclass, die auf Person zugreifen kann, geändert werden.

Zeile 14 und 15 „public“: Variablen sollten private sein, damit man von außerhalb der Klasse nicht darauf zugreifen und sie verändern kann.

Zeile 14 und 15 „final“ fehlt: ohne final kann man von der Klasse selbst aus darauf zugreifen und den Wert verändern

Zeile 37 und 51: Wir brauchen eine deep Copy damit man nicht indirekt über das Zugreifen und Ändern von birthDate gleichzeitig birthDate von einer bereits initialisierten Person verändert wird.

Aufgabe 4

1. Es wird IntelliJ verwendet

(a) find ich iwie nicht

(b) autoimport:

2. Drücke Alt + Enter wenn bei der Verwendung von unimportierten Klassen, statischen Methoden oder statischen fields ein Tooltip angezeigt wird. Es ist keine weitere Aktivierung notwendig. 3. Durch drücken von Alt+Enter wird die fehlende Importanweisung hinzugefügt, oder eine Liste von möglichen Quellen angezeigt und man kann sich eine aussuchen 4. Es spart Zeit, da man nicht jedes import Kommando selber schreiben muss und dadurch, dass man es sofort macht, kann man in einem „Fluss“ programmieren ohne dass man es möglicherweise am Ende vergisst und der Code nicht funktioniert.

(c) in Projekt suchen:

2. Drücke Ctrl + shift + F, schreibe einen String in das Suchfeld und achte dabei, dass „in project“ ausgewählt ist. Es ist keine weitere Aktivierung notwendig. 3. Das bewirkt, dass das gesamte Projekt nach diesem Wort durchsucht und eine Liste mit allen Ergebnissen, die das gesucht Wort enthalten, ausgegeben wird. 4. Gerade, wenn man mit sehr vielen Zeilen Code zutun hat und einen bestimmten Teil oder eine bestimmte Implementierung nicht wieder findet, kann man sie so finden. Außerdem kann man die Funktion auch verwenden, um nach älteren Versionen von bereits veränderten Bezeichnern oder ähnlichem zu suchen, um zu checken, ob sie überall ersetzt wurden.

(d) finds grad nicht

(e) extract method:

2. Markieren den Code-Abschnitt, den du extrahieren möchtest, drücke Ctrl + Alt + M und konfiguriere die Methode. Es ist keine weitere Aktivierung notwendig. 3. Mit extract method kann man ein Codefragment, das gruppiert werden kann, als eine separate Methode schreiben und den alten Code durch einen Aufruf der Methode ersetzen. 4. Dies bietet eine einfache, schnelle und handliche Methode um den Code übersichtlicher zu gestalten, wodurch er verständlicher und auch besser wartbar wird. Allgemein hilft dies dabei Stilregeln einfacher zu verwirklichen.

(f) Umbenennen von Bezeichnern:

2. Ändere den Namen eines Bezeichners und drücke dann Alt + Enter und wähle „Rename..“. Es ist keine weitere Aktivierung notwendig. 3. Dadurch werden alle „betroffenen“ Bezeichner geändert und geupdatet. 4. Es kann durchaus vorkommen, dass man Bezeichner nochmal ändert, da man z.B. einen besseren findet. Allerdings ist es gerade bei viel Code schwierig alle Bezeichner händisch zu finden und zu verbessern und nimmt zudem viel Zeit in Anspruch. Diese Probleme hat man durch diese Funktion nicht.