Aufgabe 2 (Casting): (8 Punkte)

Bestimmen Sie den Typ und das Ergebnis der folgenden Java-Ausdrücke und begründen Sie Ihre Antwort. Sollte der Ausdruck nicht typkorrekt sein, begründen Sie, worin der Fehler besteht.
Dabei seien die Variablen `x`, `y` und `z` wie folgt deklariert: `int x = 120; int y = 2; int z = 3;`

a) `2147483647 + '1'`

b) `(byte) 256 * 2147483648`

c) `"x" + y + z`

d) `x < y && true`

e) `9f / 3d`

f) `x + "y"- z`

g) `1 + 3f`

h) `x != 'x'? 2f : 1`

Aufgabe 4 (Programmierung): (7 Punkte)

Implementieren Sie einen einfachen Zinseszinz-Rechner in Java. Für die Ein-/Ausgabe soll die bereitgestellte Klasse `SimpleIO` genutzt werden. Um einen String `str1` in einem Fenster mit dem Titel[1] `str2` auszugeben, nutzen Sie `SimpleIO.output(str1, str2)`. Um einen Wert vom Typ `type` einzulesen, nutzen Sie `SimpleIO.getType(str)`, wobei `str` der Text ist, der dem Benutzer im Eingabefenster angezeigt wird. Um einen Wert vom Typ `int` einzulesen, benutzen Sie also z.B.
`SimpleIO.getInt("Bitte eine ganze Zahl eingeben")`.

Das Programm soll den Benutzer zunächst nach dem Startbetrag und dem Zinssatz in Prozent fragen. Beide Werte sollen als `double` eingelesen werden.
Anschließend soll der Benutzer gefragt werden, was er berechnen möchte. Dabei stehen ein *Ziel* (das die Dauer berechnet, bis ein Zielbetrag erreicht wird) oder die *Zeit* (die den nach einer gewissen Zeit angesparten Betrag berechnet) zur Auswahl. Der Benutzer soll also entweder *Ziel* oder *Zeit* als String eingeben. Gibt der Benutzer keines von beidem ein, soll Ihr Programm eine Fehlermeldung anzeigen und sich beenden.

Ziel Bei dieser Auswahl soll der Benutzer nach einem Zielbetrag als `double` gefragt werden. Anschließend soll das Programm *mit Hilfe einer geeigneten Schleife* berechnen, wie viele Jahre es braucht, bis der Zielbetrag erreicht oder überschritten wurde.
Das Programm soll dann eine Ausgabe der Form "Es dauert 2 Jahre bei einem Zinssatz von 5%, um von 100 auf den Betrag 110 zu sparen. Nach dieser Zeit hat man 110.25." liefern. Die Werte in diesem Beispiel sollen vom Programm natürlich durch die Benutzereingaben und die berechneten Werte ersetzt werden.

Zeit Bei dieser Auswahl soll das Programm den Benutzer nach einer Anzahl von Jahren als `int`-Wert fragen. Anschließend berechnet das Programm *mit einer geeigneten Schleife* wie viel man nach dieser Zeit gespart hat. Die Werte werden dann in der Form "Bei einem Zinssatz von 5% und einem Startbetrag von 100 hat man nach 2 Jahren 110.25 gespart." ausgegeben.

Ein Beispiellauf des Programms könnte also so aussehen:

```
Bitte geben Sie den Startbetrag ein.
100
Bitte geben Sie den Zinssatz als Prozentwert ein.
5
Bitte waehlen Sie aus:
Ziel: Berechnet die Zeit, bis ein gegebener Betrag angespart wurde.
Zeit: Berechnet den Betrag, der nach einer gegebenen Zeit angespart wurde.
Ziel
Bitte geben Sie den Zielbetrag ein.
110
Es dauert 2 Jahre bei einem Zinssatz von 5%, um von 100 auf den Betrag 110
zu sparen. Nach dieser Zeit hat man 110.25.
```

Hinweise:

- Um zwei Strings `str1` und `str2` auf Gleichheit zu testen, verwenden Sie `str1.equals(str2)` (und nicht `str1 == str2`).
- Beim Zinseszins wird jedes Jahr der aktuelle Betrag um den Zinssatz erhöht. Im obigen Beispiel wird der Betrag im ersten Jahr um 5 ($=5\% \cdot 100$) erhöht. Im zweiten Jahr dann um 5.25 ($=5\% \cdot 105$). Im dritten Jahr würden dann 5.5125 ($=5\% \cdot 110.25$) hinzu kommen.
- Legen Sie die bereitgestellte Datei `SimpleIO.java` einfach im gleichen Verzeichnis wie ihre Lösung ab. Dann findet Java diese automatisch.

[1] Sie dürfen in dieser Aufgabe immer z.B. "Zinseszinsrechner" als Titel verwenden.

Aufgabe 6 (Verifikation): (7 + 2 = 9 Punkte)

Gegeben sei folgendes Java-Programm P über den Integer-Variablen `n`, `i`, `sign` und `res`:

$\langle 0 \leq \texttt{n} \rangle$ (Vorbedingung)

```
sign = 1;
res = 0;
i = 0;
while (i < n) {
   res = -(res + sign);
   sign = -sign;
   i = i + 1;
}
```

$\langle \texttt{res} = (-1)^{\texttt{n}} \cdot \texttt{n} \rangle$ (Nachbedingung)

a) Vervollständigen Sie die folgende Verifikation des Algorithmus im Hoare-Kalkül, indem Sie die unterstrichenen Teile ergänzen. Hierbei dürfen zwei Zusicherungen nur dann direkt untereinander stehen, wenn die untere aus der oberen folgt. Hinter einer Programmanweisung darf nur eine Zusicherung stehen, wenn dies aus einer Regel des Hoare-Kalküls folgt.

Hinweise:

- Sie dürfen beliebig viele Zusicherungs-Zeilen ergänzen oder streichen. In der Musterlösung werden allerdings genau die angegebenen Zusicherungen benutzt.
- Bedenken Sie, dass die Regeln des Kalküls syntaktisch sind, weshalb Sie semantische Änderungen (beispielsweise von $x+1 = y+1$ zu $x = y$) nur unter Zuhilfenahme der Konsequenzregeln vornehmen dürfen.

$\langle 0 \leq \texttt{n} \rangle$

$\langle$ _____ $\rangle$

```
sign = 1;
```

$\langle$ _____ $\rangle$

```
res = 0;
```

$\langle$ _____ $\rangle$

```
i = 0;
```

$\langle$ _____ $\rangle$

```
while (i < n) {
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
   res = -(res + sign);
```

$\langle$ _____ $\rangle$

```
   sign = -sign;
```

$\langle$ _____ $\rangle$

```
   i = i + 1;
```

$\langle$_____$\rangle$

```
}
```

$\langle$_____$\rangle$
$\langle \texttt{res} = (-1)^{\texttt{n}} \cdot \texttt{n} \rangle$

b) Untersuchen Sie den Algorithmus P auf seine Terminierung. Für einen Beweis der Terminierung muss eine Variante angegeben werden und mit Hilfe des Hoare-Kalküls die Terminierung bewiesen werden.

Aufgabe 8 (Verifikation): (7 + 2 = 9 Punkte)

Gegeben sei folgendes Java-Programm P über den Integer-Variablen `n`, `i` und `res`:

$\langle 0 \leq \texttt{n} \rangle$ (Vorbedingung)

```
i = 0;
res = 0;
while (i < n) {
   if (i % 2 == 0) {
      res = res + n;
   } else {
      res = res - 1;
   }
   i = i + 1;
}
```

$\langle \texttt{res} = \lceil \frac{\texttt{n}}{2} \rceil \cdot \texttt{n} - \lfloor \frac{\texttt{n}}{2} \rfloor \rangle$ (Nachbedingung)

Vervollständigen Sie die folgende Verifikation des Algorithmus im Hoare-Kalkül, indem Sie die unterstrichenen Teile ergänzen. Hierbei dürfen zwei Zusicherungen nur dann direkt untereinander stehen, wenn die untere aus der oberen folgt. Hinter einer Programmanweisung darf nur eine Zusicherung stehen, wenn dies aus einer Regel des Hoare-Kalküls folgt.

Hinweise:

- Sie dürfen beliebig viele Zusicherungs-Zeilen ergänzen oder streichen. In der Musterlösung werden allerdings genau die angegebenen Zusicherungen benutzt.
- Bedenken Sie, dass die Regeln des Kalküls syntaktisch sind, weshalb Sie semantische Änderungen (beispielsweise von $x + 1 = y + 1$ zu $x = y$) nur unter Zuhilfenahme der Konsequenzregeln vornehmen dürfen.
- $\lceil x \rceil$ ist die kleinste Zahl $n \in \mathbb{Z}$, sodass $n \geq x$ gilt. $\lfloor x \rfloor$ ist die größte Zahl $n \in \mathbb{Z}$, sodass $n \leq x$ gilt.
- Für nicht-negative Zahlen berechnet der Java-Operator `%` die modulo-Funktion.

$\langle 0 \leq \texttt{n} \rangle$

$\langle$ _____ $\rangle$

```
i = 0;
```

$\langle$ _____ $\rangle$

```
res = 0;
```

$\langle$ _____ $\rangle$

```
while (i < n) {
```

$\langle$ _____ $\rangle$

```
    if (i % 2 == 0) {
```

$\langle$ _____ $\rangle$

$\langle$ _____ $\rangle$

```
        res = res + n;
```

$\langle$ _____ $\rangle$

```
    } else {
```

$\langle$_____$\rangle$

$\langle$_____$\rangle$

```
        res = res - 1;
```

$\langle$_____$\rangle$

```
    }
```

$\langle$_____$\rangle$

```
    i = i + 1;
```

$\langle$_____$\rangle$

```
}
```

$\langle$_____$\rangle$

$\langle \texttt{res} = \lceil \frac{\texttt{n}}{2} \rceil \cdot \texttt{n} - \lfloor \frac{\texttt{n}}{2} \rfloor \rangle$

Aufgabe 9 (Deck 1): (Codescape)

Lösen Sie die Räume von Deck 1 des Spiels Codescape.
Ihre Lösung für Räume dieses Codescape Decks wird nur dann für die Zulassung gezählt, wenn Sie die Lösung bis Montag, den 04.11.2019, um 12:00 Uhr abschicken.