Objektorientiertes Programmieren

1 Reiskocher

Implementieren Sie folgende Klassen:

Reis

- Mit einem Integer-Attribut für die Kochzeit.
- Mit einer virtuellen Funktion cook(): Soll so viele Zeilen mit der Nachricht `"Still cooking for x minutes"` ausgeben, wie der Wert der Kochzeit ist, x soll die noch verbleibende Kochzeit sein, die bei jeder Nachricht um 1 verringert wird. Hat also ein Reis eine Kochzeit von 5 soll folgende Ausgabe erscheinen:

```
"Still cooking for 5 minutes"
"Still cooking for 4 minutes"
"Still cooking for 3 minutes"
"Still cooking for 2 minutes"
"Still cooking for 1 minutes"
```

Wildreis

- Wildreis erbt von Reis.
- Hat eine Kochzeit von 10.
- Überlädt die Funktion cook(): Zuerst wird die cook-Funktion der Elternklasse aufgerufen, dann wird eine Meldung ausgegeben, dass der Wildreis fertig gekocht wurde.

Sushireis

- Sushireis erbt von Reis.
- Hat eine Kochzeit von 5.

- Überlädt die Funktion cook(): Zuerst wird die cook-Funktion der Elternklasse aufgerufen, dann wird gefragt wie viele Minuten man den Reis auskühlen lassen will (1 bis 9 Minuten). Für jede Minute soll eine Meldung `"Cooling for x minutes"` ausgegeben werden, die nach dem selben Schema funktioniert wie die Kochmeldung. Danach wird eine Meldung ausgegeben, dass der Sushireis fertig gekocht wurde.

Reiskocher

- Mit einem Reis-Pointer Attribut (Standardwert soll NULL sein).
- Mit einer Funktion chooseRice(): Wenn bereits eine Reissorte gewählt wurde, soll eine entsprechende Meldung ausgegeben und die Funktion beendet werden. Ansonsten soll es möglich sein zwischen Wildreis und Sushireis zu wählen, ein Objekt auf dem Heap zu erstellen und dessen Adresse im Reis-Pointer zu speichern.
- Mit einer Funktion cookRice(): Die Funktion soll zuerst prüfen, ob sich ein Reis im Reiskocher befindet, wurde kein Reis gewählt soll eine entsprechende Meldung ausgegeben und die Funktion beendet werden. Ansonsten wird die cook-Funktion des gewählten Reis-Objekts aufgerufen.

Sie können Ihr Programm mit folgender main-Funktion testen:

```
int main()
{
   Cooker cooker;
   cooker.cookRice();
   cooker.chooseRice();
   cooker.chooseRice();
   cooker.cookRice();
   return 0;
}
```

2 Trainingsübungen

Implementieren Sie folgende Klassen:

Mensch

- Mit zwei Integer-Attributen, Oberkörper- und Beinkraft und einem Pointer auf eine Übung (soll standardmäßig NULL sein).
- Mit einem Konstruktor der zwei Integer als Parameter erhält, die Oberkörper- und Beinkraft belegen.
- Hat eine Funktion chooseExercise(): Ist der Pointer auf die Übung bereits belegt wird der Speicher freigegeben und danach gefragt welche Übung man wählen will (Bankdrücken oder Kniebeugen) und mit wie viel Gewicht man die Übung durchführen will. Nach den Eingaben wird der Pointer mit einem neuen Objekt belegt, bei dessen Konstruktoraufruf das Gewicht mitgegeben wird.
- Hat eine Funktion train() die einen booleschen Wert zurückgibt: Ist der Pointer auf eine Übung NULL wird eine entsprechende Meldung ausgegeben und dann false zurückgegeben. Ansonsten wird die train()-Funktion des Übungsobjekts aufgerufen und danach die Werte von Oberkörper- und Beinkraft ausgegeben.
  Zum Abschluss wird noch gefragt ob man weiter machen will. Falls ja wird true zurückgegeben, ansonsten false.

Übung

- Mit einem Integer-Attribut für das Gewicht.
- Mit einer rein virtuellen Funktion train, die eine Referenz auf einen Menschen als Parameter übernimmt und einen booleschen Wert zurückgibt.

Bankdrücken

- Bankdrücken erbt von Übung.
- Hat einen Konstruktor der einen Integer-Parameter übernimmt und dessen Wert für das Gewicht-Attribut verwendet.
- Implementiert die rein virtuelle Funktion train: Ist das Gewicht kleiner als die Oberkörperkraft wird eine Meldung ausgeben, dass die Übung geschafft wurde und die Oberkörperkraft des Menschen um 1 erhöht, dann wird true zurückgegeben. Ist das Gewicht kleiner als die doppelte Oberkörperkraft wird eine Meldung ausgegeben, dass die Übung gerade noch geschafft wurde und die Oberkörperkraft des Menschen um 3 erhöht, dann wird true zurückgegeben. Ansonsten wird eine Meldung ausgegeben, dass die Übung unmöglich ist und false zurückgegeben.

Kniebeugen

- Kniebeugen erbt von Übung.
- Hat einen Konstruktor der einen Integer-Parameter übernimmt und dessen Wert für das Gewicht-Attribut verwendet.
- Implementiert die rein virtuelle Funktion train: Ist das Gewicht kleiner als die Unterkörperkraft wird eine Meldung ausgeben, dass die Übung geschafft wurde und die Unterkörperkraft des Menschen um 1 erhöht, dann wird true zurückgegeben. Ist das Gewicht kleiner als die doppelte Unterkörperkraft wird eine Meldung ausgegeben, dass die Übung gerade noch geschafft wurde und die Unterkörperkraft des Menschen um 3 erhöht, dann wird true zurückgegeben. Ansonsten wird eine Meldung ausgegeben, dass die Übung unmöglich ist und false zurückgegeben.

Sie können Ihr Programm mit folgender main-Funktion testen:

```
int main()
{
   Human human(50, 30);
   human.train();
   human.chooseExercise();
   while(human.train())
      human.chooseExercise();
   return 0;
}
```

3 Benutzersammlung

Implementieren Sie folgende Klasse:

Sammlung

- Hat eine Map mit Integer-Schlüsseln und String-Werten.
- Hat eine Funktion addUser, die einen Integer und String als Parameter akzeptiert: Fügt einen neuen Eintrag in der Map ein, wobei die übergebenen Parameter Schlüssel und Wert sein sollen.
  Probieren Sie dabei sowohl das Einfügen über den [ ]-Operator als auch über die insert-Funktion aus und beobachten Sie das Verhalten des Programms.
- Hat eine Funktion printUsers(): Geht durch die ganze Map und gibt alle Schlüssel-Wert-Paare aus.
- Hat eine Funktion getUser, die einen Integer als Parameter übernimmt und einen String retourniert: Prüft ob ein Eintrag mit dem übergebenen Integer als Schlüssel in der Map existiert. Falls ja wird der entsprechende Wert zurückgegeben, ansonsten wird eine Meldung zurückgegeben, dass der Schlüssel nicht gefunden wurde.

Sie können Ihr Programm mit folgender main-Funktion testen:

```
int main()
{
   Collection collection;
   collection.addUser(5, "Mr. A");
   collection.addUser(7, "Tiffany");
   collection.printUsers();
   cout << collection.getUser(3) << endl;
   collection.addUser(7, "Smith");
   collection.printUsers();
   collection.addUser(3, "Smith");
   cout << collection.getUser(3) << endl;
}
```