Aufgaben für Tag 3

Arraylist

1. Erstellt eine Arraylist in der ihr eure 2 Lieblingsfächer eintragt. Oh, sorry: Es sollten doch 3 sein.
   Durchlauft nun die Arraylist mit einer for-Schleife und gebt die Fächer in der Form aus: "Meine Lieblingsfächer in der Schule waren:... ." Ändert nun das erste Fach in Sport.

2. Die Lottozahlen am Samstag waren 1,17,25,45,38,10. Speichert sie in der Arraylist 'lottozahlen'.
   Prüft ob sich die Zahl 12 in der Arraylist befindet. Nun das Gleiche für die 45. Lasst euch nun die Länge der Liste ausgeben.
   Angenommen, ihr würdet diesen Befehl eingeben:
   lottozahlen.remove(1);
   Was glaubt ihr: Welche Zahl würde entfernt werden. Probiert es erst aus, nachdem ihr eine Vermutung aufgestellt habt.

Vererbung

1. Ihr verwaltet in einem Programm die Mitarbeiter einer Firma.
   Alle Mitarbeiter haben einen Vornamen, einen Nachnamen und eine Personalnummer. Zudem wird allen Mitarbeitern ein Gehalt ausgezahlt. Alle diese Attribute sind natürlich *private* und verfügen über entsprechende Getter und Setter. Sie werden dem Konstruktor der Klasse übergeben.
   Die Mitarbeiter im Außendienst fahren zudem einen Dienstwagen einer bestimmten Marke (fahreMitDienstwagen() SOUT: "Mitarbeiter *nachname* fährt einen Wagen der Marke *marke*"). Man kann weiterhin abfragen wieviele Außenkontakte sie hatten (*int* getAussenkontakte()). Alle zusätzlichen Attribute sollen ebenfalls dem Konstruktor üebergeben werden.
   Mitarbeiter in der Managementetage bekommen einmal jährlich einen Bonus ausgezahlt (zahleBonus(*double* bonus) SOUT: "Sie bekommen einen Bonus von *bonus* und kaufen sich ein schickes Haus"), sie sind zudem in der Lage die Personalnummer eines jeden anderen Mitarbeiters herauszufinden.
   Hierfür benötigen sie nur den Nachnamen des Mitarbeiters (holePersonalnummer(*String* nachname)). Die EDV ist jedoch noch nicht fertig und antwortet in jedem Fall einfach nur : "Mitarbeiter *nachname* ist nicht verfügbar".
   Mitarbeiter in der Lagerverwaltung haben keine besonderen Merkmale, die sie von anderen Mitarbeitern unterscheiden.

Sehr großes (End-)Projekt

Erstellt eine Klasse "Rennschnecke"

1. Rennschnecken sollen folgende Eigenschaften (Objektvariablen) besitzen:
   - einen Namen
   - eine Rasse
   - eine Maximalgeschwindigkeit
   - die Schnecke soll wissen welchen Weg sie bereits zurückgelegt hat
2. Erstelle für die Klasse Rennschecke einen Konstruktor, der den Instanzvariablen beim Erstellen einer neuen Instanz Werte zuweist.
3. Lege in der Klasse Rennschnecke eine Methode *krieche()* an, welche die Schnecke abhängig von ihrer Maximalgeschwindigkeit eine zufällige Strecke weiter bewegt. Soll heißen: Sie kriecht eine zufällige Strecke größer null und kleiner ihrer Maximalgeschwindigkeit.
   - Tipp: Schaut euch die Methode *Math.random()* aus der Java API an.
4. Lege in der Klasse Rennschnecke eine Methode *public String toString()* an, welche die Daten der Schnecke mit return als String zurückgibt.
5. Teste deine Klasse, indem du probehalber ein Rennschneckenobjekt erzeugst und seine Daten auf der Konsole ausgibst.
   - Verwende zum Ausgeben der Daten die *toString()* Methode der Rennschnecke.

Erstellt eine Klasse "Rennen"

1. Ein Rennen hat folgende Eigenschaften:
   - einen Namen
   - die Anzahl der teilnehmenden Schnecken
   - die teilnehmenden Schnecken selbst (z.B. in einer ArrayList)
   - die Länge der zu kriechende Strecke
2. Überlege dir, welche dieser Werte bereits im Konstruktor gesetzt werden sollten.
3. Lege in der Klasse Rennen eine Methode *void addRennschnecke(Rennschnecke neueSchnecke)* an, welche dem Rennen eine Schnecke hinzufügt.
4. Lege in der Klasse Rennen eine Methode *void removeRennschnecke(String name* an, welche eine Schnecke aus dem Rennen entfernt.
5. Lege in der Klasse Rennen eine Methode *public String toString()* an, welche die Daten des Rennens mit return als String zurückgibt.
   - Tipp: Um die daten der beteiligten Schnecken zurückzugeben, könnt ihr deren *toString()* Funktion benutzen.

6. Teste deine Klasse vom Hauptprogramm aus!

7. Lege in der Klasse Rennen eine Methode *Rennschnecke ermittleGewinner()* an, welche **null** zurückliefert, wenn noch keine der teilnehmenden Schnecken das Ziel errericht hat und anderenfalls die Gewindnerschnecke zurückgibt.

8. Lege in der Klasse Rennen eine Methode *void lasseSchneckenKriechen()* an, welche alle teilnehmenden Schnecken einmal kriechen lässt.

9. Lege in der Klasse Rennen eine Methode *void durchfuehren()* an, welche so lange *lasseSchneckenKriechen()* aufruft, bis eine der Schnecken das Ziel erreicht hat.

    - Tipp. Ob eine Schnecke im Ziel angekommen ist, kannst du mit der Methode *ermittleGewinner()* herausfinden.

Erstellt eine Klasse "Wettbuero"

1. Ein Wettbuero hat folgende Eigenschaften:

    - Es weiß, für welches Rennen es seine Wetten entgegennimmt.
    - Es verfügt über eine Liste (z.B. eine ArrayList) von angenommenen Wetten.
    - Es hat einen festen Faktor, mit welchem Wetteinsätze bei einem Gewinn multipliziert werden.

2. Lege in der Klasse Wettbuero eine Methode *wetteAnnehmen(String schneckenName, int wettEinsatz, String spieler)* an, welche eine Wette entgegennimmt. Die Wette ist bezogen auf eine Schnecke für das Rennen, das von dem Büro bearbeitet wird.

    - Um die einzelnen Wetten speichern zu können, sollten ihre Daten in eigenen Objekte der Klasse Wette gespeichert werden. Erstellt euch diese Klasse selbst.