Wiederholungsaufgaben abstrakte Klassen und Interfaces

Konstruktoren

Aufgabe 1)

Erstellt eine abstrakte Klasse Ticket, die eine Methode public float calcTicketPrice() enthält. Dabei soll die Klasse mindestens die Objektvariablen Preis, Eventname, Ort und Datum haben.

Erstellt einen Konstruktor, der die Variable *price* übergeben bekommt. Erstellt einen zweiten Konstruktor der sowohl *price* als auch *eventName* übergeben bekommt. Ein dritter Konstruktor soll sowohl *price* als auch *eventName*, *date* und *place* übergeben bekommen. Dabei sollen sich die Konstruktoren aufrufen mit this(…);

Erstellt mehrere Objekte der Klasse Ticket und ruft dabei die verschiedenen Konstruktoren auf.

Abstrakte Klassen

Aufgabe 1)

Erstellt eine abstrakte Klasse Ticket, die eine abstrakte Methode public float calcTicketPrice() enthält.

Erstellt eine CinemaTicket, die von der Klasse Ticket erbt und die Methode calcTicketPrice() überschreibt. Der Preis einer Kinokarte berechnet sich aus einem Basispreis und einem Aufschlag um 3 Euro, wenn der Film länger als 3 Stunden ist.

Erstellt eine Klasse ConzertTicket, die ebenfalls von der Klasse Ticket erbt und die Methode calcTicketPrice() überschreibt. Der Preis einer Konzertkarte berechnet sich aus einem Basispreis und einem Aufschlag abhängig von der Sitzplatzreihe (Basispreis/Sitzreihe).

Überlegt euch selbstständig was für zusätzliche Objektvariablen/Instanzvariablen, Methoden und Konstruktoren für diese Klassen notwendig sind.

Erstellt ein Array* vom Typ Ticket mit je 2 Objekten des Typs CinemaTickets und ConzertTicket.

Aufgabe 2)

Erstellt eine abstrakte Klasse Animal, die eine abstrakte Methode public String getFeed() enthält. Dabei soll die Methode getFeed() null oder „nothing to eat“ zurückgeben, wenn ein Tier nichts zu fressen hat. Ansonsten soll das zurückgegeben werden was bei den Tieren aktuell auf dem Speiseplan steht.

Erstellt eine Klasse Lion, die die Methode getFeed() überschreibt. Löwen jagen, wenn es dunkel ist und wenn Wild (engl. Game) in der Nähe ist. Ihr könnt die Methode in diesem Fall beliebig kompliziert machen oder einfach mit zwei boolean Variablen für dunkel und Wild in der Nähe. Nimmt die schwierige, wenn ihr richtig ins Programmieren reinkommen wollt und ihr mittlerweile ein ganz gutes Verständnis für abstrakte Klassen habt😊

Erstellt eine Klasse CommonBlackBird (Amsel), die ebenfalls die Methode getFeed() überschreibt. Eine Amsel frisst normalerweise Insekten, wenn keine da sind aber auch Beeren.

Überlegt euch selbstständig was für zusätzliche Objektvariablen/Instanzvariablen, Methoden und Konstruktoren für diese Klassen notwendig sind.

Erstellt ein Array* vom Typ Animal mit je 2 Objekten des Typs Lion und CommonBlackBird.

Interfaces

Aufgabe 1)

Erstellt ein Interface IMovable mit der Methode void move().

Erstellt eine Klasse Ball, die das Interface IMovable implementiert und die Methode move() überschreibt. Ein Ball hat eine x und y Koordinate und bewegt sich mit einer Geschwindigkeit in x Richtung (xVelocity) und mit einer Geschwindigkeit in y Richtung (yVelocity).

Erstellt eine Klasse Car, die das Interface IMovable implementiert und die Methode move() überschreibt. Ein Auto hat eine x und y Koordinate und bewegt sich mit einer Geschwindigkeit in x Richtung (xVelocity).

Überlegt euch selbstständig was für zusätzliche Objektvariablen/Instanzvariablen, Methoden und Konstruktoren für diese Klassen notwendig sind.

Erstellt ein Array* vom Typ IMovable mit je 2 Objekten des Typs Ball und Car.

Aufgabe 2) (Interface und abstrakte Klassen)

Erstellt ein Interface ISound mit der Methode void makeSound().

Erweitert die Klasse CommonBlackBird aus der Aufgabe 2 abstrakte Klasse und zusätzlich implementiert das Interface ISound. Es gibt die Möglichkeit entweder eine Ausgabe auf der Konsole zu machen oder wenn ihr es cool lösen wollt, könnt ihr auch einen Sound abspielen. Ein Beispiel wie man einen Sound spielt, findet ihr unter 02Vorlesung\playSound.

Erstell eine Klasse Jellyfish, die ebenfalls von der Klasse Animal erbt. Überschreibt die Methode getFeed(). Eine Qualle kann keine Geräusche erzeugen, daher soll Jellyfish das Interface ISound nicht implementieren.

Erstellt anschließend ein Array* vom Typ Animal mit je 2 Objekten von CommonBlackBird und Jellyfish und ruft die getFeed()-Methode auf. Wenn das jeweilige Objekt auch vom Typ ISound ist, soll auch die Methode makeSound() aufgerufen werden.

*) Ihr könnt auch eine ArrayList verwenden, um dem Umgang damit zu üben 😊