

Wiederholklausur

Modul	Programmierung II
Lehreinheit	Fortgeschrittene Programmierung
Dozent	Daniel Appenmaier
Kurse	WWIBE121 und WWIBE221
Zeitraum	Q1 2023
Matrikelnummer	

Aufgabe	Max. Punkte	Punkte
1	12	
2	12	
3	16	
4	12	

Hinweise zu den Klassendiagrammen

- Der Stereotyp **record** impliziert, dass die Datenklasse einen entsprechenden Konstruktor, Getter zu allen Attributen sowie entsprechende Implementierungen für die Object-Methoden besitzt
- Der Stereotyp **enumeration** impliziert, dass die Aufzählung einen passenden, privaten Konstruktor sowie ggbf. passende Setter und Getter besitzt

Allgemeine Hinweise

- Pakete und Klassenimporte müssen nicht angegeben werden
- In Aufgabe 1 muss nur eine der beiden Optionen bearbeitet werden

Viel Erfolg!

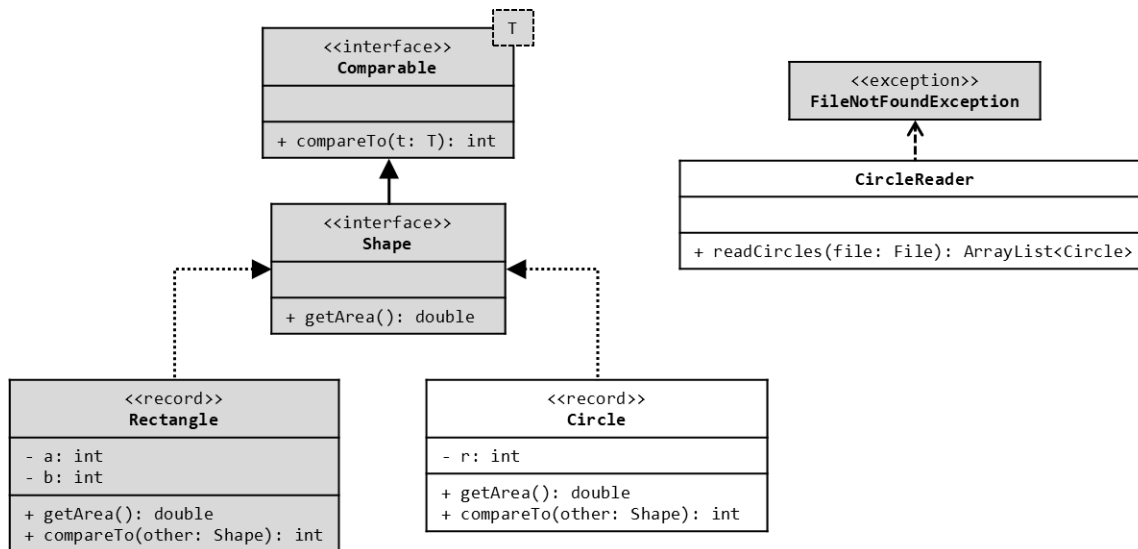
Aufgabe 1 – Option 1 (12 Punkte)

- Erläutere kurz, zu was man unter den Begriffen **Upcast**, **Downcast** und **Polymorphie** versteht (3 Punkte)
- Erläutere kurz, was man unter der **Catch-or-Throw Regel** versteht (2 Punkte)
- Erläutere kurz, wofür der **Wildcard-Typ** „?“ benötigt wird (2 Punkte)
- Erläutere kurz den wesentlichen Unterschied zwischen **intermediären** und **terminalen Operationen** und benenne eine beliebige **intermediäre Operation** (3 Punkte)
- Erläutere kurz, welchen Ansatz die **Testgetriebene Entwicklung** verfolgt (2 Punkte)

Aufgabe 1 – Option 2 (12 Punkte)

Erstelle die Klassen **Circle** (5 Punkte) und **CircleReader** (7 Punkte) anhand des abgebildeten Klassendiagramms.

Klassendiagramm



Hinweise zur Klasse **Circle**

- Die Methode **double** `getArea()` soll den Flächeninhalt des Kreises zurückgeben
- Die Methode **int** `compareTo(other: Shape)` soll so implementiert werden, dass damit Kreise aufsteigend nach ihrem Flächeninhalt sortiert werden können

Hinweis zur Klasse **CircleReader**

Die Methode **ArrayList<Circle>** `readCircles(file: File)` soll eine Liste mit allen Kreisen der eingehenden Kreisdatei zurückgeben.

Allgemeine Hinweise

Die Konstante **PI** der Klasse **Math** beinhaltet den Wert der Kreiszahl Pi.

Beispielhafter Aufbau der Kreisdatei

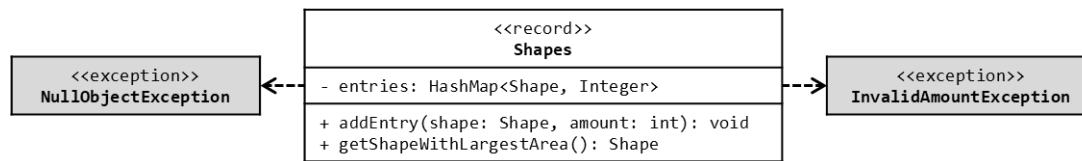
```

3
5
...
  
```

Aufgabe 2 (12 Punkte)

Erstelle die Klasse **Shapes** anhand des abgebildeten Klassendiagramms.

Klassendiagramm



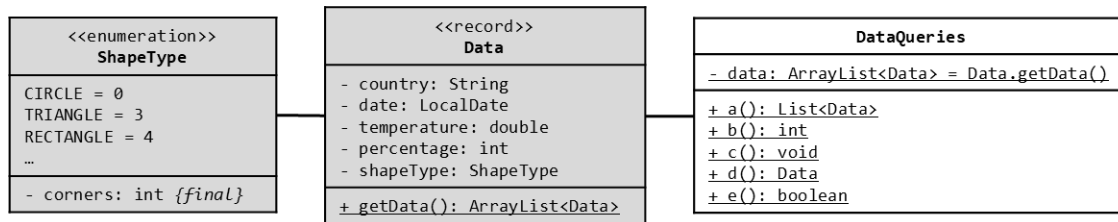
Hinweise zur Klasse **Shapes**

- Die Methode **void addEntry(shape: Shape, amount: int)** soll den Einträgen (**entries**) die eingehende geometrische Form sowie die eingehende Menge hinzufügen bzw. ersetzen. Für den Fall, dass die eingehende geometrische Figur der Null-Referenz entspricht, soll die Ausnahme **NullPointerException** ausgelöst werden und für den Fall, dass die eingehende Menge einen Wert kleiner gleich Null besitzt, die Ausnahme **InvalidAmountException**
- Die Methode **Shape getShapeWithLargestArea()** soll die geometrische Form mit dem größten Flächeninhalt zurückgeben

Aufgabe 3 (16 Punkte)

Erstelle die Klasse **DataQueries** anhand des abgebildeten Klassendiagramms.

Klassendiagramm

Hinweise zur Klasse **DataQueries**

- Die statische Methode **List<Data> a()** soll alle Rechtecke absteigend sortiert nach der Prozentzahl zurückgeben
- Die statische Methode **int b()** soll die Summe aller Eckpunkte aller geometrischer Figuren zurückgeben, bei denen die Temperatur über 20°C liegt
- Die statische Methode **void c()** soll alle Datensätze der Monate Dezember, Januar und Februar, bei denen der Prozentwert über 50% liegt in der Form *[Land] [Jahr]: [Prozentsatz]%* ausgeben
- Die statische Methode **Data d()** soll einen beliebigen Datensatz zurückgeben, der die Bedingung *Temperatur * Prozentsatz / 100 > 10* erfüllt. Für den Fall, dass kein entsprechender Datensatz vorhanden ist, soll der Wert **null** zurückgegeben werden
- Die statische Methode **boolean e()** soll zurückgeben, ob es einen Datensatz aus China gibt, bei dem das Datum dem aktuellen Datum entspricht

Allgemeine Hinweise

- Die Methode **int getMonthValue()** der Klasse **LocalDate** gibt den Monat einer Datumsangabe als Zahlenwert zwischen 1 und 12 zurück
- Die Methode **int getYear()** der Klasse **LocalDate** gibt das Jahr einer Datumsangabe zurück
- Die statische Methode **LocalDate now()** der Klasse **LocalDate** gibt das aktuelle Datum zurück

Beispielhafte Konsolenausgabe

```

c():
Indien 2021: 69%
Brasilien 2018: 69%
USA 2021: 74%
...
  
```

Klassendiagramm



- 5

Cheatsheet

Java API

Klasse	Methode	Rückgabotyp
ArrayList<E>	add(e: E)	boolean
	add(index: int, element: E)	void
	contains(o: Object)	boolean
	forEach(action: Consumer<T>)	void
	get(index: int)	E
	remove(index: int)	E
	remove(o: Object)	boolean
	size()	int
Aufzählung	valueOf(arg0: String)	Aufzählung
	values()	Aufzählung[]
Boolean	valueOf(s: String)	Boolean
Collections	sort(list: List<T>)	void
	sort(list: List<T>, c: Comparator<T>)	void
Comparable<T>	compareTo(o: T)	int
Comparator<T>	compare(o1: T, o2: T)	int
Double	valueOf(s: String)	Double
Entry<K, V>	getKey()	K
	getValue()	V
HashMap<K, V>	containsKey(key: Object)	boolean
	containsValue(value: Object)	boolean
	entrySet()	Set<Entry<K, V>>
	get(key Object)	V
	keySet()	Set<K>
	put(key: K, value: V)	V
	values()	Collection<V>
Integer	valueOf(s: String)	Integer
List<E>	<u>of(elements: E...)</u>	List<E>
Object	equals(object: Object)	boolean
	toString()	String
Optional<T>	get()	T
	isPresent()	boolean
	orElse(other: T)	T
Random	nextInt(bound: int)	int
String	charAt(index: int)	char
	length()	int
	split(regex: String)	String[]
	toLowerCase()	String
	toUpperCase()	String

JUnit 5

Klasse	Methode	Rückgabotyp
Assertions	assertEquals(expected: Object, actual: Object)	void
	assertNull(actual: Object)	void
	assertSame(expected: Object, actual: Object)	void
	assertThrows(expectedType: Class<T>, executable: Executable)	T
	assertTrue(condition: boolean)	void
Executable	execute()	void

Java Stream API

Klasse	Methode	Rückgabotyp
Collectors	<u>toList()</u>	Collector<T, ?, List<T>>
	<u>groupingBy(classifier: Function<T, K>)</u>	Collector<T, ?, Map<K, List<T>>>
Consumer<T>	accept(t: T)	void
DoubleStream	average()	OptionalDouble
	sum()	double
Function<T, R>	apply(t: T)	R
IntStream	average()	OptionalDouble
	sum()	int
Predicate<T>	test(t: T)	boolean
Stream<T>	allMatch(predicate: Predicate<T>)	boolean
	anyMatch(predicate: Predicate<T>)	boolean
	collect(collector: Collector<T, A, R>)	R
	count()	long
	distinct()	Stream<T>
	filter(predicate: Predicate<T>)	Stream<T>
	findAny()	Optional<T>
	findFirst()	Optional<T>
	forEach(action: Consumer<T>)	void
	limit(maxSize: long)	Stream<T>
	map(mapper: Function<T, R>)	Stream<R>
	mapToDouble(mapper: ToDoubleFunction<T, R>)	DoubleStream
	mapToInt(mapper: ToIntFunction<T, R>)	IntStream
	max(comparator: Comparator<T>)	Optional<T>
	min(comparator: Comparator<T>)	Optional<T>
	skip(n: long)	Stream<T>
	sorted(comparator: Comparator<T>)	Stream<T>
ToDoubleFunction<T, R>	applyAsDouble(value: T)	double
ToIntFunction<T, R>	applyAsInt(value: T)	int