Wintersemester 2019/2020		Zahl der Blätter: Blatt Nr:	21 1
Fachbereich:	Informationstechnik	Semester:	SWB/TIB/2
Prüfungsfach:	OOS 1	Prüfungsnr:.	1052027
Hilfsmittel:	keine elektronischen Hilfsmittel	Zeit:	90 min
Name:		Matrikel-Nr.:	

Hinweis: Der auf den Blättern jeweils freigelassene Raum reicht im Allgemeinen vollständig für die stichwortartige Beantwortung der Fragen, bzw. für die Lösungen aus. Tragen Sie daher auf **jedem** Blatt Ihren Namen und Ihre Matrikelnummer ein und nutzen Sie diese Blätter zur Abgabe Ihrer Antworten und Lösungen.

Aufgabe 1: Allgemeine Fragen (ca. 20 Min.)

Aufgabe 1.1

Bitte beurteilen Sie die folgenden allgemeinen Aussagen. Machen Sie jeweils ein Kreuzchen in der Spalte „wahr" oder „falsch". Begründen Sie jeweils Ihre Wahl.

Aussage	wahr	falsch
Mit **class** A { **int** a; **friend** B; }; kann die Klasse A auf die privaten Elemente der Klasse B zugreifen. **Begründung:** Anderstum		X
Es gibt keinen Unterschied zwischen einer Zuweisung und einer Initialisierung. -> = (↓ ==) **Begründung:** Zuweisung vergleicht, Initialisierung setzt	X	
Für eine tiefe Kopie benötigt man einen selbstdefinierten Kopierkonstruktor. **Begründung:** nein, der Kopierkonstruktor erstellt nur flache Kopie		X

Aussage	wahr	falsch
Der „**<<“ Operator** kann nur als Freundfunktion oder als globale Funktion überladen werden **Begründung:**	X	
Bei class Dreieck { Punkt x, y, z; }; spricht man von einer Aggregation. **Begründung:** Nein, es sind Kompositione		X

Aufgabe 1.2

Bitte beantworten Sie die jeweiligen Fragen.

Frage 1:
Was ist der Unterschied zwischen einem Pointer und einer Referenz?

Antwort:

Referenz: gibt die Adresse &

Pointer: erstellt eine Kopie *

Frage 2:
Welche Konstruktoren gibt es, wie viele Parameter und welche Funktion haben sie?

Antwort:

Kopier,

Default,

Parametisierter,

Frage 3:

Was ist das grundsätzliche Problem bei der Mehrfachvererbung und wie kann dieses gelöst werden?

Antwort:

Frage 4:

Was ist ein String-Stream und was bietet dieser für Vorteile?

Antwort:

Frage 5:

Was ist der Unterschied zwischen statischer Bindung und dynamischer Bindung?

Antwort:

Aufgabe 2: Klassendeklaration (ca. 20 Min.)

ACHTUNG: Lesen Sie die folgende Aufgabe KOMPLETT durch, bevor Sie mit der Lösung beginnen. Bitte trennen Sie **Deklaration** und **Implementierung** der Klassen.

Die Teilaufgaben von Aufgabe 2 können auch unabhängig voneinander gelöst werden.

Im Folgenden sollen nun Klassen für ein Projektmanagement definiert werden. Folgende Abbildung zeigt das dazugehörende Klassendiagramm.

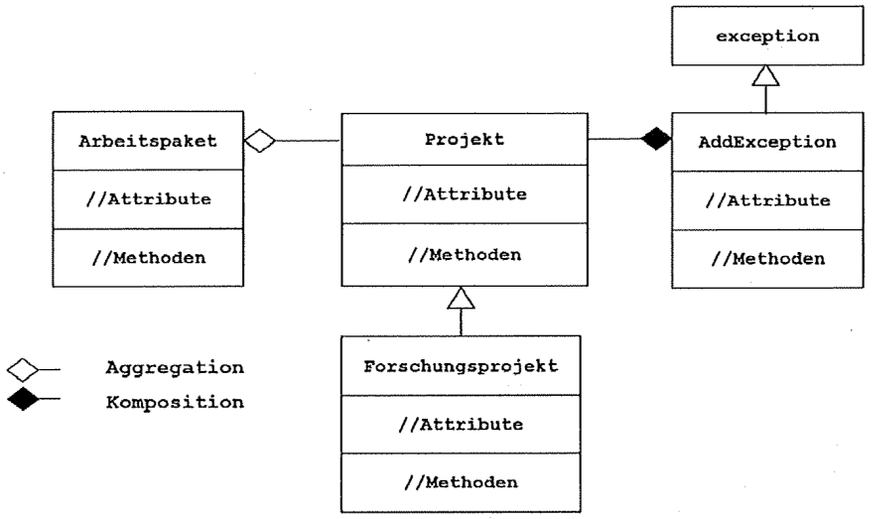

Abbildung 1: Klassendiagramm – Projektmanagement

2.1 Eine Klasse **Arbeitspaket**, soll mit folgenden Elementen definiert werden.

a) eine konstante Instanzvariable **id** vom Typ **int**, die eine eindeutige ID für das Arbeitspaket festhält,

b) eine Instanzvariable **name** vom Typ **string**, die den Namen des Arbeitspakets festhält,

c) einen Instanzvariable **personentage** vom Typ **float**, die den Umfang der Personentage für dieses Arbeitspaket festhält,

d) einen Konstruktor, der die **ID**, den **Namen** und die **Personentage** übergeben bekommt,

e) eine Instanzmethode **getPersonentage**, die die **Personentage** des Arbeitspaketes **zurückgibt**,

f) einen **== Operator,** mit dem zwei **Arbeitspakete** anhand ihrer **id** verglichen werden können (z.B. arbeitspaket1 == arbeistpaket2),

2.2 Eine Klasse **AddException**, die von der C++-Klasse exception erbt und folgende Eigenschaften hat,

a) eine Instanzvariable **text** vom Typ **string**, die den Text für den Fehler enthält,

b) einen Konstruktor, der den **Text** für die **AddException** übergeben bekommt,

c) eine Redefinition der **what-Metode**, die den **Text** der **AddExeption** auf der Konsole ausgibt.

2.3 Eine Klasse **Projekt**, die allgemeine Eigenschaften und das Verhalten für ein Projekt in sich vereint.

a) eine konstante Instanzvariable **id** vom Typ **int**, die die ID des Projektes festhält,

b) eine Instanzvariable **name** vom Typ **string**, die den Namen des Projektes festhält,

c) eine Instanzvariable **kapazität** (Personentage) vom Typ **float**, die die Kapazität des gesamten Projektes angibt:

d) ein **VectorArray** mit dem Namen **abs**, der Pointer auf **Arbeitspakete** verwaltet, die zu dem entsprechenden Projekt gehören,

e) einen **Konstruktor**, der die **ID,** den **Namen** und die **Kapazität** des Projektes übergeben bekommt,

f) einen **Kopier-Konstruktor**, um ein Projekt zu kopieren,

g) eine Instanzmethode **getProjektauslasung**, die die **Projektauslastung** (Summe aller Personentage der zu diesem Projekt gehörenden Arbeitspakete) des Projektes **zurückgibt**,

h) eine Instanzmethode **addArbeitpaket**, die ein Arbeitspaket zu dem Projekt hinzufügt, sofern das entsprechende Arbeitspaket noch nicht vorhanden ist und die zusätzlichen Personentage des Arbeitspakets nicht die Kapazität des Projektes „sprengt". Falls das entsprechende Arbeitspaket schon existiert, soll eine **AddException** geworfen werden mit dem **Text *„Error: Arbeitspaket schon vorhanden!".*** Falls die Kapazität des Projektes überschritten wird, soll eine **AddException** mit dem **Text *„Error: Projektkapazität erreicht!"*** geworfen werden.

2.4 Eine Klasse **Forschungsprojekt**, die von der Klasse **Projekt** erbt, soll mit folgenden Elementen definiert werden.

a) eine Instanzvariable **forschungsgebiet** vom Typ **string**, die das Forschungsgebiet festhält (z.B. „Astronomie"),

b) eine Instanzvariable **foerderer** vom Typ **string**, die den Förderer des Forschungsprojekts festhält (z.B. „Land Bayern"),

c) einen Konstruktor, der die **ID**, den **Namen** und die **Kapazität** für ein **Projekt**, als auch das **Forschungsgebiet** und den **Förderer** für das **Forschungsprojekt** übergeben bekommt,

d) einen **Kopierkonstruktor**, der ein Forschungsprojekt kopiert,

2.5 Eine Funktion `void testProjektmanagement()` mit folgenden Eigenschaften:

a) Drei Arbeitspakete mit folgenden Eigenschaften:

(1) Arbeitspaket 1: ID = 123, Name = AP1, Personentage = 3;

(2) Arbeitspaket 2: ID = 124, Name = AP2, Personentage = 5;

(3) Arbeitspaket 3: ID = 125, Name = AP3, Personentage = 4;

b) ein Forschungsprojekt mit folgenden Eigenschaften:

   (1) ID: 3349,

   (2) Name: FP1,

   (3) Kapazität: 9 Personentage,

   (4) Forschungsbereich: Astronomie

   (5) Förderer: Land Bayern

c) die angelegten Arbeitspakete 1-3, werden dem Forschungsprojekt hinzugefügt, mögliche AddExceptions beim Hinzufügen abgefangen und der Grund für die Exception auf der Konsole ausgegeben,

d) Erzeugen Sie ein neues Forschungsprojekt mittels einer Kopie aus dem vorher erstellen Forschungsprojekt und geben Sie die Projektauslastung für diese Kopie auf der Konsole aus.

Ergänzen Sie das folgende Programmgerippe und die Funktion testProjektmanagement(). Schützen Sie die Datenelemente vor Zugriffen durch klassenfremde Methoden; erlauben Sie aber abgeleiteten Klassen den Zugriff. Verwenden Sie – falls möglich – konstante Methoden.

Trennen Sie **Header** (Aufgabe 2) und **Implementierung** (Aufgabe 3).

Prototypen der Klassen (Aufgabe 2)

`//include` *(was für die Klassendeklaration der Klasse Arbeitspaket benötigt wird, falls nötig)*

```
#pragma once
#include <iostream>
#include <string>
```

Klassendeklaration der Klasse Arbeitspaket

```
class Arbeitspaket {
// Instanzvariablen
const int id;
String name;
 float personentage;


// Konstruktor
Arbeitspaket (int id, string name, float personentage);



// Methoden

 float getPersonentage() const;

bool operator== (arbeitspaket&, arbeitspaket&);
```

`//include` *(was für die Klassendeklaration der Klasse AddException benötigt wird, falls nötig)*

```
#pragma once
#include <iostream>
#include <string>
```

Klassendeklaration der Klasse AddException

```
class AddException : public exception        {
// Instanz- und Klassenvariablen
String text;



// Konstruktor
AddException ( string text);



// Methoden
const char* what() const throw override();



};
```

`//include` *(alles was für die Klassendeklaration der Klasse Projekt benötigt wird, falls nötig)*

```
#pragma once
#include <iostream>
```

Klassendeklaration der Klasse Projekt

```
class Projekt: {
// Instanz- und Klassenvariablen
const int id;
String name;
float Kapazitaet
vector <Arbeitspakete*> abs
// Konstruktor
Projekt (int id, String name, float kapazitaet);
Projekt (int id&, String name&, float kapazitaet&)



// Methoden
float getProjektauslastung();
float addArbeitPaket (Arbeitspakete* abs);



};
```

`//include` *(was für die Klassendeklaration der Klasse Forschungsprojekt benötigt wird, falls nötig)*

Klassendeklaration der Klasse Forschungsprojekt

```
class Forschungsprojekt : public Projekt        {
// Instanzvariablen
String forschungsgebiet;
String foerderer;


// Konstruktor
Forschungsprojekt( Projekt id, Projekt name, Projekt kapazitaet, string forschungsgebiet, String foerderer);
Forschungsprojekt( *& ; & , & )



};
```

Aufgabe 3: Implementierung der Methoden der Klassen (ca. 22 min)

Programmieren Sie bitte hier und auf den folgenden Seiten außerhalb der Klassen **`Arbeitspaket`**, **`AddException`**, **`Projekt`** und **`Forschungsprojekt`** die angegebenen Elemente aus:

`//include` *(was für die Klassendefinition der Klasse Arbeitspaket benötigt wird, falls nötig)*

Klassendefinition der Klasse Arbeitspaket

```
// Arbeitspaket Konstruktor

Arbeitspaket::  Arbeitspaket(int id, string name, float personentage):
                    id(id),
                    name(name),
                    personentage(personentage) {

                }

// Arbeitspaket Methoden

float Arbeitspaket:: getPersonentage(){
    return personentage;
}

bool Arbeitspaket:: operator ==( Arbeitspakete& a1 , Arbeitspaket& a2) {
    if (a1.id == a2.id) {
        return true;
        }
    else {return false;}
```

`//include` *(was für die Klassendefinition der Klasse AddException benötigt wird, falls nötig)*

Klassendefinition der Klasse AddException

```
// AddException Konstruktor
AddException(String text): text(text){

// AddException Methoden
```

`//include` *(was für die Klassendefinition der Klasse Projekt benötigt wird, falls nötig)*

Klassendefinition der Klasse Projekt

```
// Projekt Konstruktor


// Projekt Methoden
```

`//include` *(alles was für die Klassendefinition der Klasse Forschungsprojekt benötigt wird, falls nötig)*

Klassendefinition der Klasse Forschungsprojekt

```
//Forschungsprojekt Konstruktor
```

Implementierung der Methode testProjektmanagement

```
void testProjektmanagement () {

// Erzeugen der 3 Arbeitspakete




// Erzeugen des Forschungsprojektes




// Fügen Sie die Arbeitspakete dem Forschungsprojekt hinzu, fangen Sie ggf.
eine AddException ab (mit Ausgabe des Grundes auf der Konsole)









// Erstellen Sie eine Kopie des oben erzeugen Forschungsprojektes und geben Sie
die Projektauslastung der Kopie aus
```

Aufgabe 4: Programmausgabe (ca. 11 min)

Analysieren Sie das nachfolgende Programm und schreiben Sie die Ausgabe des Programms unterhalb von „//**Ausgabe:**“.

```
#include <iostream>
#include <string>
using namespace std;

class A {
public:
    virtual void f() { cout << "A::f()->"; }
    void f(int i) { cout << "A::f(" << i << ")->"; }
    void g() { cout << "A::g()" << endl; }
    void g() const { cout << "A::g() const" << endl; }
    void h() { f(); f(1); g(); }
    void h() const { g(); }
};
class B : public A {
public:
    void f() { cout << "B::f()->"; }
    void f(int i) { cout << "B::f(" << i << ")->"; }
    virtual void g() const { cout << "B::g() const" << endl; }
    void h() { f(); f(2); g(); }
    void h() const { g(); }
};
class C : public B {
public:
    void f() { cout << "C::f()->"; }
    virtual void f(int i) { cout << "C::f(" << i << ")->"; }
    void g() { cout << "C::g()" << endl; }
    void g() const { cout << "C::g() const" << endl; }
    void h() { f(); f(3); g(); }
    void h() const { g(); }
};
void main() {
      C o_c; A * p_c = &o_c;
      B o_b; A * p_b = &o_b;
      A o_a; A * p_a = &o_a;
      const C o_c_const;
      const A * p_c_const = &o_c_const;
      o_c.h();
      o_b.h();
      o_a.h();
      o_c_const.h();
      p_c->h();
      p_b->h();
      p_a->h();
      p_c_const->h();
}
```

```
//Ausgabe Aufgabe 4:
```

Aufgabe 5: Objektorientierung und Polymorphie (ca. 15 Min.)

Gegeben ist das folgende Programmfragment:

```
int main(){

      Pizza* mista = new Pizza("Mista");
      mista->belegtMit("Salami");
      mista->belegtMit("Pilzen");

      Gericht* hawaii = new Pizza("Hawaii");
      hawaii->belegtMit("Schinken");
      hawaii->belegtMit("Ananas");

      Burger* hamburger = new Burger("Hamburger");
      hamburger->belegtMit("Hackfleisch");

      Gericht* cheesburger = new Burger("Cheesburger");
      cheesburger->belegtMit("Hackfleisch");
      cheesburger->belegtMit("Käse");

      array<Gericht*, 4> speisekarte = {mista, hawaii, hamburger, cheesburger};

      for (Gericht* g : speisekarte) {
            g->zubereitung();   //*[1]
      }

    return 0;
}
```

Geben Sie die entsprechenden Implementierungen für die Klassen *Gericht*, *Pizza* und *Burger* an, sodass das oben vorhandene Programmfragment kompiliert werden und mittels der Zeile **[1]* folgendes auf der Konsole ausgegeben werden kann:

```
Pizza Mista. Pizzaboden, belegt mit:
- Salami
- Pilzen
Pizza Hawaii. Pizzaboden, belegt mit:
- Schinken
- Ananas
Hamburger. Brötchen mit:
- Hackfleisch
Cheeseburger. Brötchen mit:
- Hackfleisch
- Käse
```

Die Ausgabe soll durch die Auswertung des zuvor zusammengesetzten Objektgeflechtes entstehen. Nutzen Sie die Vererbung von Instanzvariablen und Instanzmethoden entsprechend. (Eine **Trennung** von Deklaration und Implementierung ist **NICHT** notwendig):

Implementierung Aufgabe 5 (Methode) 0.5P

Sommersemester 2019		Blatt Nr:	21 / 21
Prüfungsfach:	OOS 1	Prüfungsnr:.	1052027
Name:		Matrikel-Nr.:	