CHALMERS
EXAMINATION / TENTAMEN

Course code/kurskod	Course name/kursnamn			
DIT 042/043	Object oriented Programming			
Anonymous code Anonym kod		Examination date Tentamensdatum	Number of pages Antal blad	Grade Betyg
272		16/08 -2022	11	4

* I confirm that I've no mobile or other similar electronic equipment available during the examination.
Jag intygar att jag inte har mobiltelefon eller annan liknande elektronisk utrustning tillgänglig under eximinationen.

Solved task Behandlade uppgifter No/nr		Points per task Poäng på uppgiften	Observe: Areas with bold contour are to completed by the teacher. Anmärkning: Rutor inom bred kontur ifylles av lärare.
1	X	15	
2	X	20	
3	X	35	
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16			
17			
Bonus poäng			
Total examination points Summa poäng på tentamen		70	

CHALMERS | Anonymous code: DIT043 | Anonym kod: 272 | Points for question (to be filled in by teacher): Poäng på uppgiften (ifylles av lärare): 15 | Consecutive page no. Löpande sid nr 1. | Question no. Uppgift nr Q1

Q1.

1. "Constructor: total: 20 ✓
2. "Constructor: total: 24" ✓
3. "Constructor: total: 27 ✓
4. "mA has n1 = 3 ✓ n2 = -3 ✓
5. "mB has n1 = 5 ✓ n2 = 10 ✓
6. "mC has n1 = 4 ✓ n2 = 2 ✓
7. "method: n1 = 13 ✓ n2 = 3 ✓ total = 43 ✓
8. "method: n1 = 13 ✓ n2 = 5 ✓ total = 63 ✓
9. "main: n1 = 5 ✓ n2 = 10 ✓ total = 63 ✓
10. "mA has n1 = 2 ✓ n2 = 4 ✓
11. "mC has n1 = 10 ✓ n2 = 4 ✓
12. "mA has n1 = 10 ✓ n2 = 4 ✓
13. "Constructor: total: 58 ✓
14. "mA has n1 = -10 ✓ n2 = -5 ✓
15. "mC has n1 = 5 ✓ n2 = 10 ✓

15p

2

Q2.

```
public class CourseResult {

private List<CourseResult> result;

private final String courseName;
private final int grade;
private final double credit;


public CourseResult(String courseName, int grade,
 double credit, List<CourseResult> result) throws Exeption {
  if(grade<0 || grade>100){
    throw new Exeption("Invalid course result
                        data.")
  }
    if(credit<=0.0){
      throw new Exeption("Invalid course result data.")
  }
    if(courseName.isEmpty()){
      throw new Exeption("Invalid course result data.")

    }
    this.result = new ArrayList<CourseResult>();
    this.courseName = courseName;
    this.grade = grade;
    this.credit = credit;
}
```

? why this attribute? - low cohesion.

5p

2p

Answer only one question on this page. Do not write on the back of this paper

CHALMERS | Anonymous code: DIT042 | Anonym kod: 272 | Points for question (to be filled in by teacher): 0 | Poäng på uppgiften (ifylles av lärare): 0 | Consecutive page no. Löpande sid nr 3 | Question no. Uppgift nr Q2

```
@Override
public String toString(){
    return "("+this.courseName + "," + this.credit + "," +
    this.grade+")";
}

Public String getCourseName() { return this.courseName; }
Public int getGrade(){ return this.grade; }
public double getCredit() { return this.credit; }

}
```

1p (accounted for on previous page).

→ equals?

0p

2

Q2.2

```
public void printTopTwoResults() throws Exeption {
for(CourseResult courseResult: result){
    if(result.size()<2){
        throw new Exeption("Cannot print. Student
        has less than two
        course results.")
}
    int topGrade1= 0;
    int topGrade2 = 0;
for(CourseResult courseResult: result){
    if(courseResult.getGrade()> topGrade1){
        topGrade1 = courseResult.getGrade;
    if(courseResult.getGrade<topGrade1){
        topGrade2 = courseResult.getGrade;
    System.out.println("Best grades: " +
topGrade1.getCourseName + "and" +
topGrade2.getcourseName;
}
```

Not needed

topG2 should then become topG1. s/d

else?

top grade 2 will always change then.

printing should be outside the loop.

2p

2

Q2.3

One advantage of using immutable objects is that it's highly secure, because we cannot change it. Changing objects could lead to inconsistency and risks of breaking the code. We prevent any changes, and unwanted behaviours.

bp

A disadvantage is however the fact that we're unable to change it. Sometimes we need to change objects, but if its immutable we'll simply have to create a new one. More coding. more more coding. It's more execution time

Also an advantage is that anything coupled with it wont break, since we cannot change it.

Q2.4 Benefits of encapsulation is that we make our code more secure. ✓ With encapsulation the programmer decides who and how the attributes are accessed and changed. (getters and setters).

In the student class we have a getter for the id, so we can get the data, but not a setter, meaning we cannot change it. ✓ It brings security to the code since we cannot change id from anywhere.

However, a setter is implemented for the fullName of a student. If a student changes their name, the name change will be possible.

We won't get access to the private attributes without these.

1. Controll access → Those are the same benefits. Missing second one.
2. Reduce risk of unwanted changes

5p

2

Q2.5 We need to use the class CourseResult as a reference variable via composition? to connect the two classes and then , store it in a file. Put CourseResult in the constructor.

why input if we aim to save?

2p

Then we need to create a file output and input stream. By that we make it possible to deflate and reverse it back to its normal state.

→ Serializable?

CHALMERS | Anonymous code: DITG432 | Anonym kod: 272 | Points for question (to be filled in by teacher): Poäng på uppgiften (ifylles av lärare): 12 | Consecutive page no. Löpande sid nr 8 | Question no. Uppgift nr Q3.1

Q3.1 1. Hendy Lamar : Points = 40. Credits : 2512.0

2. Mary Keller : Points = 0. Credits : 1518.0

3. Ada Lovelace : Points = 210. Credit : 2194.0

4. Dorothy Vaughan : Points = 50. Credits : 1415.4

12p

CHALMERS | Anonymous code DIT043 | Anonym kod 272 | Points for question (to be filled in by teacher) | Poäng på uppgiften (ifylles av lärare) 10 | Consecutive page no. Löpande sid nr 9 | Question no. Uppgift nr 3.2

Q3.2 Since we are using inheritance we have polymorphic benefits. SilverMembership and GoldMembership extends from Membership. That means all of the attributes and methods are inherited and therefore we can code less. * The methods and attributes that are implemented and or overritten is polymorphism. ? We use whatever methods that exist and override them to make them perform something different.

* Since we code less due to inheritance, the code is easy to read through and the design is great. However, any changes in the parent-class will affect the child, so both design and execution is affected. 10p

Overriding also affects both design and execution, but mainly execution since its risky to change methods.

A third benefit is that with polymorphism we create new things from already excisting ones. Example we can extend from the suitable class and create more memberships. We inhent the same code and change it.

Isn't that already described in your benefit 1?

2

Q3.3

We could change it to an abstract class and do abstract methods. We would then inherit the attributes and methods, but we would have to code more.

We reduce the risk of overriding if the methods don't have a body. However, I don't see the need to do an abstract class here, because we cannot instanziate it, so then we would have to do the abstract class, then a regular membership that inherits from the abstract.

Since most methods are already shared as it is now, and making an abstract class wouldn't have that many benefits in this specific case, I don't argee with Sams suggestion.

Though, we can have methods with a body if it is something that is gonna be used by all classes and therefor not override. Abstract is a safer way to do inheritance but as mentioned, I don't think we need it this time.

8p

CHALMERS	Anonymous code: DITO43 / Anonym kod: 272	Points for question (to be filled in by teacher): / Poäng på uppgiften (ifylles av lärare): 5	Consecutive page no. Löpande sid nr 11 / Question no. Uppgift nr Q3.4

Q3.4

Changing the visibility from protected to private would make the code even more encapsulated. Protected allows access/changes to the attributes if they are in the same hirarcy tree. Private will prevent that without getters and setters, so the encapsulation is higher. It could introduce bugs without getters and setters, but if we use getters and setters I don't see why there would be any bugs.

5p