OOP BONUS PARTY

NAME 4 OOP CONCEPTS AND EXPLAIN THEM

EXPLAIN THE DIFFERENCE BETWEEN ABSTRACTION AND ENCAPSULATION IN ELEGANT WAY

Main OOP concepts

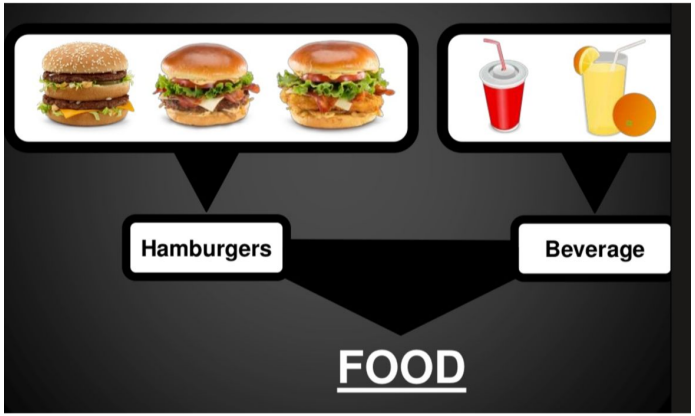
1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphisam

Abstraction

Hiding non-essential features and showing the essential features

(or)
Hiding unnecessary details from the users

Example:
TV Remote Button (you don't see the circuits in it, right?)



Encapsulation

Encapsulation is packaging the data and functions into a single component.

It allows selective hiding of properties and methods in an object by building an wall to protect the code from accidental corruption.

Abstraction is about expressing external simplicity and

Encapsulation is about hiding internal complexity

> *If I have seen further it is by standing on the shoulders of Giants.*
> *- Isaac Newton*

nanos gigantum humeris insidentes

NAME AND EXPLAIN THE OOP PRINCIPLE BEHIND THESE WORDS (2 pts)

REUSE

by following methodologies/code created by other people (or by us before), we are able to achieve great results without “reinventing the wheel” and doing many trial and errors.

«На плеча́х гига́нтов»,— крылатая фраза и развёрнутая метафора, фразеологизм, обозначающий самую общую формулу преемственности в познании, науке или искусстве: «новые достижения с опорой на открытия предыдущих деятелей».

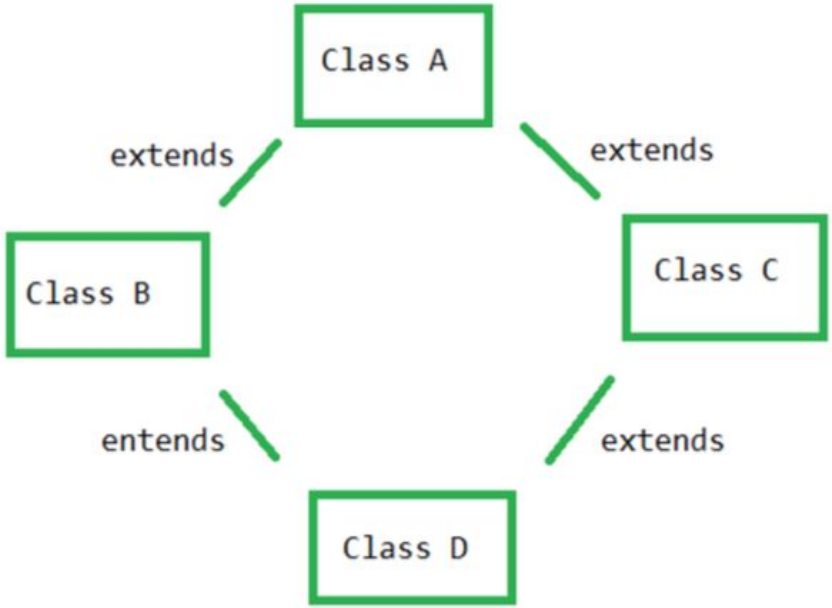
What is a default interface method? I told you not to misuse the default methods. Why?

NAME 2 TYPES OF INHERITANCE

Types of Inheritance:

1. Multiple inheritance.
2. Multilevel inheritance.

"What do you do when you have multiple common base classes in the different superclasses?



Diamond problem:

In multiple inheritance there is every chance of multiple properties of multiple objects with the same name available to the sub class object with same priorities leads for the ambiguity.

Virtual vs Abstract methods

1813

An abstract function cannot have functionality. You're basically saying, any child class MUST give their own version of this method, however it's too general to even try to implement in the parent class.

A virtual function, is basically saying look, here's the functionality that may or may not be good enough for the child class. So if it is good enough, use this method, if not, then override me, and provide your own functionality.

```
public abstract class myBase
{
    //If you derive from this class you must implement this method.
    public abstract void YouMustImplement();

    //If you derive from this class you can change the behavior but
    public virtual void YouCanOverride()
    {
    }
}
```

```
public abstract class Shape
{
   public abstract void Paint(Graphics g, Rectangle r);
}
public class Ellipse: Shape
{
   public override void Paint(Graphics g, Rectangle r) {
      g.DrawEllipse(r);
   }
}
public class Box: Shape
{
   public override void Paint(Graphics g, Rectangle r) {
      g.DrawRect(r);
   }
}
```

Is there any sense in a virtual method without a body?

```
public virtual void CountX(){}
```

Is there any sense in a virtual method without a body?

Almost it is a rare occasion, yes. Because if the default behavior is to do nothing, but derived classes might want to do something. It's a perfectly valid structure.

If these methods (Pre, Post) were abstract, then all derived classes would need to implement them

```
abstract class Base
  {
      public void ProcessMessages(IMessage[] messages)
      {
          PreProcess(messages);

          // Process.

          PostProcess(messages);
      }

      public virtual void PreProcess(IMessage[] messages)
      {
          // Base class does nothing.
      }

      public virtual void PostProcess(IMessage[] messages)
      {
          // Base class does nothing.
      }
  }

  class Derived : Base
  {
      public override void PostProcess(IMessage[] messages)
      {
          // Do something, log or whatever.
      }

      // Don't want to bother with pre-process.
  }
```

Inheritance exhibits the “IS-A” relationship. Name the remaining relationships

You can prevent method overriding by using final keyword. Why you might need finalizing implementations? Example?

Why preventing method overriding?

You use final when you don't want subclass changing the logic of your method by overriding it due to security reason.

This is why String class is final in Java. This concept is also used in template design pattern where template method is made final to prevent overriding.

EXPLAIN THE PURPOSE OF TRICKY CLONEABLE INTERFACE IN JAVA. WHY METHOD CLONE() IS SITUATED IN AN OBJECT CLASS ?

EXPLAIN THE DIFFERENCE BETWEEN IMPORTING A CLASS and EXTENDING A CLASS

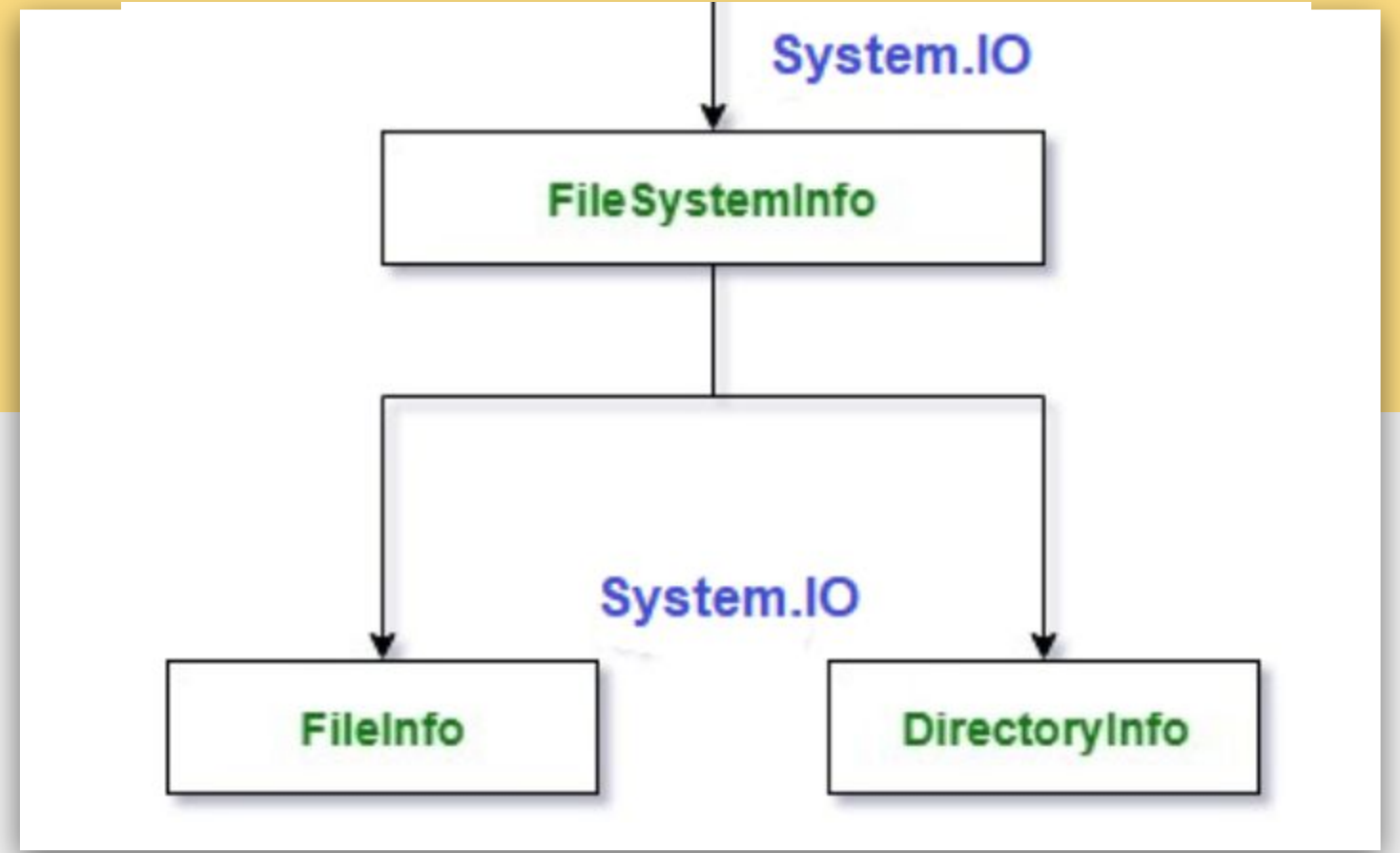
Painter class can be left unchanged while new types of Objects in Paint are added. Why?

```
public class Painter {
   private List<Paintable> paintableObjects;
   public Painter(){
     paintableObjects = new ArrayList<Paintable>(); }
   public void paintAllObjects(){
      for(Paintable paintable : paintableObjects){
         paintable.paint();  } }}
public interface Paintable {
    public void paint();}
```

SPELL OUT THE CODE FOR A GENERIC MAX METHOD

```
public class Max
{
  // Return the maximum between two objects
 public static Comparable max(Comparable o1, Comparable o2)
  {
    if (o1.compareTo(o2) > 0)
      return o1;
    else
      return o2;
  }

 }
```

PROPOSE A DESIGN FOR CLASSES TO REPRESENT FILES AND DIRECTORIES ?


System.IO
FileSystemInfo
System.IO
FileInfo
DirectoryInfo

What is boxing and unboxing?

1. Why should boxing be avoided? (Choose one.)
A. It adds overhead.
B. Users must have administrative privileges to run the application.
C. It makes code less readable.

How Java achieves platform independence?

Explain the connection between an Iphone and PowerSocket here

```
Interface Pluggable {}
Class PowerSocket{
        boolean charge(Pluggable p){

}
Class iPhone extends Device implements Pluggable
```

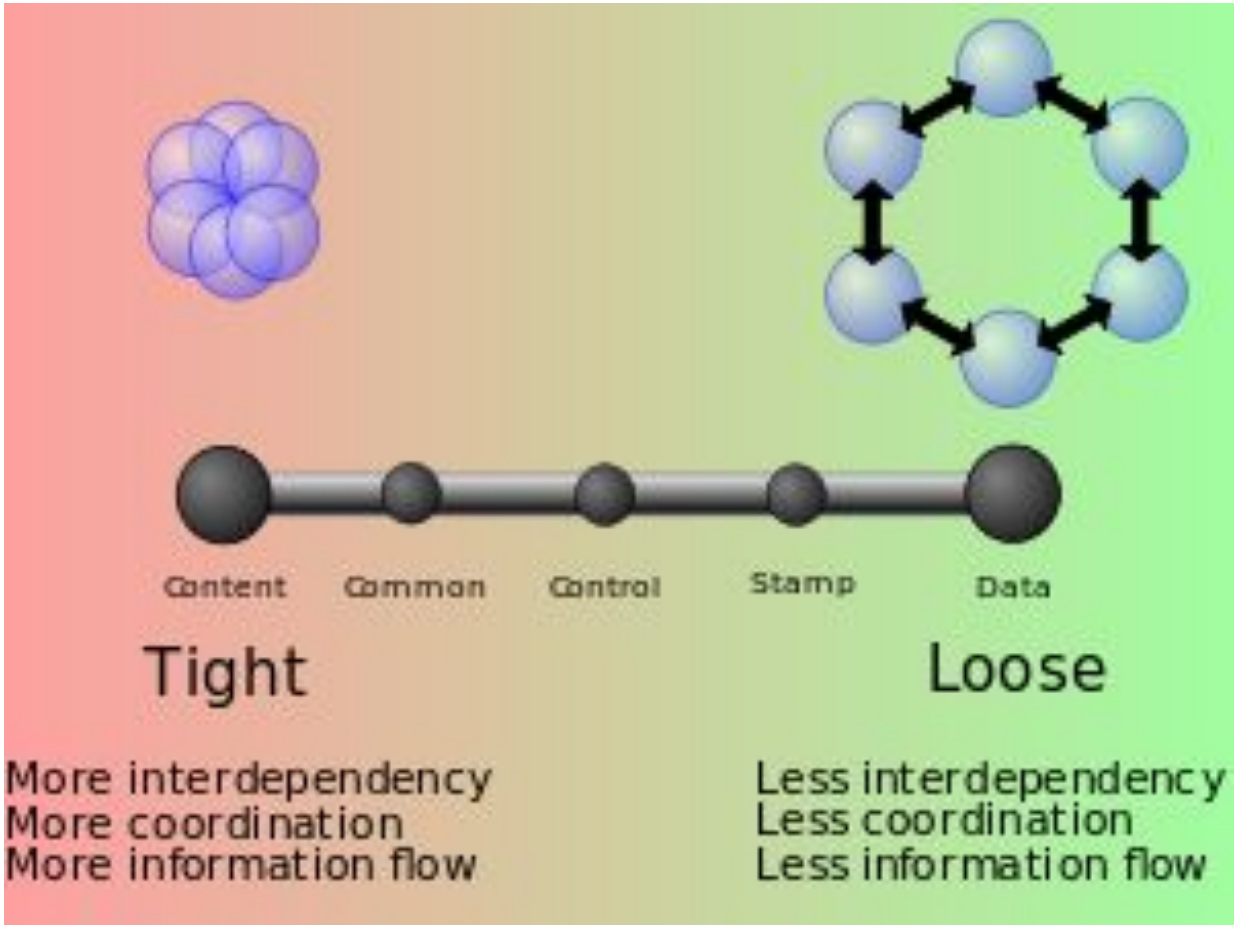
Name *ALL* similarities and differences between abstract class and interface as a song (3 pts). Должна быть римфа!

On the example of a ***Flight*** class, give example of (2 pts) :

Final static field
Final non-static field
Static non-final field
Non-static Non-final field

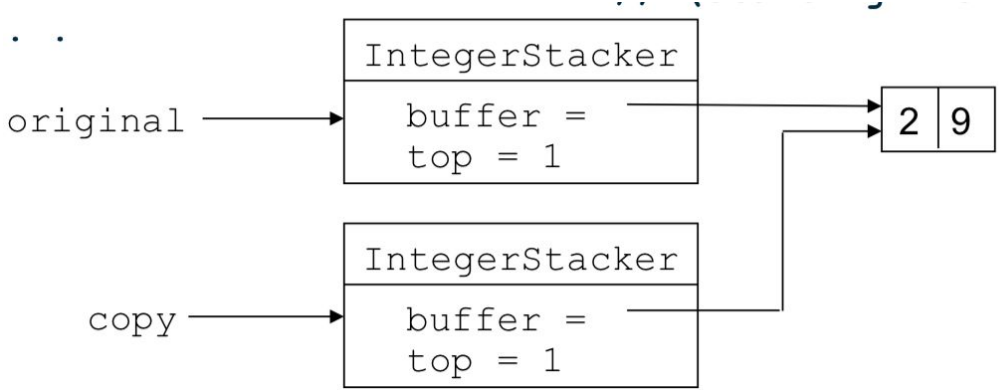
Final static field - airlines name (e.g., Air Astana)
Final non-static field - departure city
Static non-final field - Meal types (raw, kids menu, vegan, gluten-free, etc.), Menu
Non-static Non-final field - id or code

Explain: Low coupling. E.g., components are loosely coupled.


Content
Common
Control
Stamp
Data
Tight
Loose
More interdependency
More coordination
More information flow
Less interdependency
Less coordination
Less information flow

Explain: High cohesion

Explain, what can you see here?



Explain: Frozen specifications

Explain: God Object Antipattern

main() method in Java is static. Why?

The execution of Java program starts with public static void main(String args[ ]), also called the main() method.

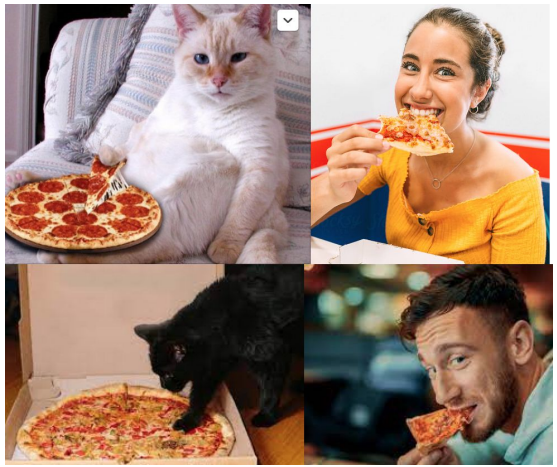
static: The keyword indicates the variable, or the method is a class method. The method main() is made static so that it can be accessed without creating the instance of the class. When the method main() is not made static, the compiler throws an error because the **main() is called by the JVM before any objects are made**, and only static methods can be directly invoked via the class.

Why sometimes we may meet some class having 2 methods that do the same thing, but have different names (2 points)?

Explain, why Interfaces never contain instance fields?

Describe in details the design situation we had during our interface lecture: watching user transcripts

Why CanHavePizza cannot be an abstract class (1 point)? Why not just having eatPizza() method in a cat and Person without having an interface (1 point)?



How does the throw keyword differ from the throws keyword?

While the throws keyword allows declaring an exception, the throw keyword is used to explicitly throw an exception.

Polymorphism

```
Class Restaurant {
      boolean servePizza(CanHavePizza eater){
            eater.eatPizza();
        if(eater instance of Person)
                  // process payments
}
}
```

delpapa.servePizza(new Cat());

На клумбе растет 40 цветов, за ними непрерывно следят два садовника и поливают увядшие цветы, при этом оба садовника очень боятся полить одни и тот же цветок. There are 40 flowers growing on the flowerbed, two gardeners are constantly following them and watering the wilted flowers, while both gardeners are very afraid to water the same flower (3 points).

Which collection will you use for that task and describe all modifiers this collection will have.

Есть 3 курильщика, 3 ученых и посредник. Курильщик непрерывно скручивает сигареты и курит их. Чтобы скрутить сигарету, нужны табак, бумага и спички. Ученый пишет статьи, для этого нужны бумага и ручка и спичка (чтобы зажечь свечу и работать). У одного курильщика есть табак, у второго - бумага, а у третьего - спички. У ученых тоже по одному предмету. Посредник кладет на стол по два разных случайных компонента. Тот курильщик или тот ученый, у которого есть третий компонент, забирает компоненты со стола, скручивает сигарету и курит либо пишет статью. Посредник дожидается, пока курильщик /ученый закончит, затем процесс повторяется. Создавая приложение, моделирующее поведение курильщиков, ученых и посредника, какие ООП принципы / темы вы бы использовали?

There are 3 smokers, 3 scientists and one mediator. The smoker continuously rolls cigarettes and smokes them. To roll a cigarette, you need a plate, paper and matches. A scientist writes articles, for this he needs paper, a pen and a match (to light a candle and work). One smoker has a plate, the second one has paper, and the third one has matches. The scientists also have one item. The mediator puts two different random components on the table. A smoker or a scientists who has a third component, takes the component from the table, rolls a cigarette and smokes or writes an article. The mediator waits until the smoker/scholar finishes, then the process repeats. When creating an application that models the behavior of smokers, scientists, and mediators, what OOP principles / topics would you use?

В тихом городке есть парикмахерская. Парикмахер всю жизнь обслуживает посетителей. Он очень любит животных, так что у него есть целых 3 кошки, которые любят поспать, но иногда следят за стрижкой. Когда в салоне никого нет, сам парикмахер спит в кресле. Когда посетитель приходит и видит спящего парикмахера, он будет его, садится в кресло и спит, пока парикмахер занят стрижкой. Если посетитель приходит, а парикмахер занят, то он встает в очередь и засыпает. Если есть ожидающие посетители, то парикмахер будит одного из них, и ждет пока тот сядет в кресло парикмахера и начинает стрижку. Когда посетитель, парикмахер или кошка спят, их уровень энергии повышается. Какие ООП принципы / темы вы бы использовали здесь?

There is a hairdresser in a quiet town. The hairdresser serves visitors all his life. He loves animals very much, so he has 3 cats who like to sleep, but sometimes they watch him for a haircut. When there is no one in the salon, the hairdresser himself sleeps in the chair. When a visitor comes and sees a sleeping hairdresser, he wakes him up, sits in a chair and sleeps while the hairdresser is busy cutting hair. If a visitor comes and the hairdresser is busy, he will stand in line and fall asleep. If there are waiting visitors, the hairdresser wakes up one of them and waits until he sits in the hairdresser's chair and starts cutting. When a visitor, a hairdresser or a cat sleeps, their energy level rises. Which OOP principles / topics would you use here?

```
interface Educated {
        int PUPIL = 0;
        int BACHELOR = 1;
        int MASTER = 2;
        int PHD = 3;
        void setEducationLevel (int level);
        int getEducationLevel();
}
```

Serialization converts the object state to serial bytes. How can you make some data non-serialized during this process and why we might need it?

```
transient int price;
```

transient **is a Java keyword which marks a member variable not to be serialized.**

Propose an OOP design for a class representing a Polynomial (2 points).

The coefficients of the polynomial should be passed as an array parameter with array type double in the constructor of your class. the resulting Polynomial is defined by:

```
coefficients[0] + coefficients[1] * x +
coefficients[2] * x2 + ... + coefficients[n] *
xn
```

Why equals takes Object as a parameter?

Explain the connection between equals and hashcode? Why we need hashcode if we have equals?

Is it a good idea to have *equals* and *hashcode* in the interface *MyCollection*?

Are equals=true and compareTo()=0 are identical?
Provide example!

Comparator and Comparable. Name 3 differences

Explain the example from lecture (1 point)
What was the name of the App? (1 point)

```
Interface Game {
        a
        b
        c
}
Interface IGame extends Game {
        d
}
Class MemoryGame  implements IGame {}
Class LogicGame implements Game{}
Class App {
        void getStatistics (Game g){}
}
```

```
class Employee extends Person implements Vegetarian{}
```

Does it make any sense to you?
Not all employees in the world are vegetarians, right. But…

Explain 2 reasons to extend interface from the other interface? 1 point for each reason. Provide examples along with the explanations.

When you extend interface, and the only difference in methods of super and sub interfaces is in the return type, then what will be?

There will be a compile-time error

Explain two types of typecasting ?

What exactly is the difference between a variable and an object in Java?

A variable is a name that you give something. An object is a piece of memory that represents an instance of a class. Usually this means "an object is a thing."

So let's say you have a Customer class. Some different combinations:

```
1  // one variable, no objects
2  Customer firstCustomer = null;
3
4  // one variable, one object
5  Customer cust = new Customer();
6
7  // two variables, one object (if you change a, it changes b)
8  Customer a = new Customer();
9  Customer b = a;
10
11 // one variable, 10 objects
12 Customer[] customers =
   callMethodThatGivesMeAnArrayOfNCustomers(10);
```

We all know that we use super to access parent methods. Give example when parent refers to a kid class method (3 points)

ANIMAL class :

```
public final String voice(int n){
    return voice().repeat(n);
}
public abstract String voice() ;
```

Guess what is happening there (2 pts)

For example, given citations = [3, 0, 6, 1, 5], which means the researcher has 5 papers in total and each of them had received 3, 0, 6, 1, 5 citations respectively.

```
public int [redacted](int[] citations) {
    Arrays.sort(citations);

    int result = 0;
    for(int i=citations.length-1; i>=0; i--){
        int cnt = citations.length-i;
        if(citations[i]>=cnt){
            result = cnt;
        }else{
            break;
        }
    }

    return result;
}
```

Pakizar Shamoi

ПОДПИСКА ОФОРМЛЕНА

Professor, KBTU
Подтвержден адрес электронной почты в домене kbtu.kz
fuzzy sets and logic artificial intelligence soft computing color theory computational aesthetics

НАЗВАНИЕ	ПРОЦИТИРОВАНО	ГОД
Modeling aesthetic preferences: Color coordination and fuzzy sets P Shamoi, A Inoue, H Kawanaka Fuzzy Sets and Systems 395, 217-234	11	2020
Fuzzy model for human color perception and its application in e-commerce P Shamoi, A Inoue, H Kawanaka International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems …	10	2016
Perceptual color space: Motivations, methodology, applications P Shamoi, A Inoue, H Kawanaka 2014 Joint 7th International Conference on Soft Computing and Intelligent …	8	2014
Fuzzy color space for apparel coordination P Shamoi, A Inoue, H Kawanaka Open Journal of Information Systems (OJIS) 1 (2), 20-28	8	2014
FHSI: Toward more human-consistent color representation P Shamoi, A Inoue, H Kawanaka Journal of Advanced Computational Intelligence and Intelligent Informatics …	7	2016
Deep color semantics for e-commerce content-based image retrieval P Shamoi, A Inoue, H Kawanaka Proceedings of the 2015 International Conference on Fuzzy Logic in …	7	2015
Computing with words for direct marketing support system P Shamoi, A Inoue Midwest Artificial Intelligence and Cognitive Science Conference	4	2012
Fuzzification of HSI Color Space and its Use in Apparel Coordination. P Shamoi, A Inoue, H Kawanaka MAICS, 11-17	2	2014

idx	0	1	2	3	4	5
	0	1	3	5	5	6

i=5, count = 1, citation = 6, 6 >=1, update h-index ↓1, continue

i=4, count = 2, citation = 5, 5 >=2, update h-index ↓2, continue

i=3, count=3, citation =5, 5 >=3, update h-index ↓3, continue

i= 2, count=4, citation =3, 3 ~~>=~~ 4, break.

Result = 3.

```
public int hIndex(int[] citations) {
    Arrays.sort(citations);

    int result = 0;
    for(int i=citations.length-1; i>=0; i--){
        int cnt = citations.length-i;
        if(citations[i]>=cnt){
            result = cnt;
        }else{
            break;
        }
    }

    return result;
}
```

Identify which oops concept were used in below scenario:

”In a group of 5 boys one boy never give any contribution when group goes for eating out, party or anything.
Suddenly one beautiful girl joins the same group and then the aforementioned boy spends lots of money for the group.”

Identify which oops concept is used in below scenario:

”In a group of 5 boys one boy never give any contribution when group goes for eating out, party or anything.
Suddenly one beautiful girl joins the same group and then the aforementioned boy spends lots of money for the group.”

Answer:
Runtime Polymorphism

Is there any sense in a constructor in abstract class?

What are two ways to handle exceptions in java?

Is it possible to write multiple catch blocks under a single try block? Describe the approach

Is there any sense in a private constructor ?

Singleton class

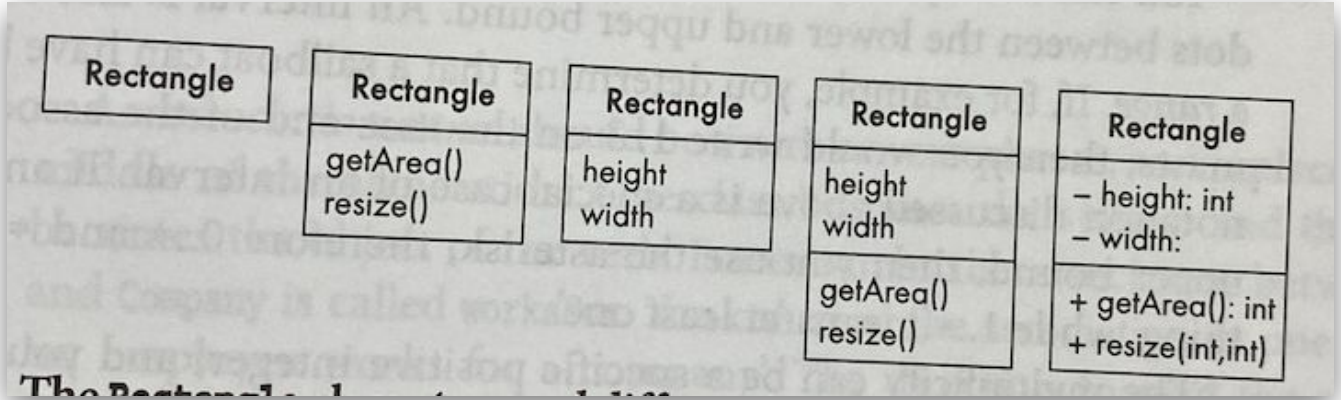
Answer:
Sometimes, yes! E.g., for a Singleton pattern.

In software engineering, the singleton pattern is a software design pattern that restricts the instantiation of a class to one object. This is useful when exactly one object is needed to coordinate actions across the system.

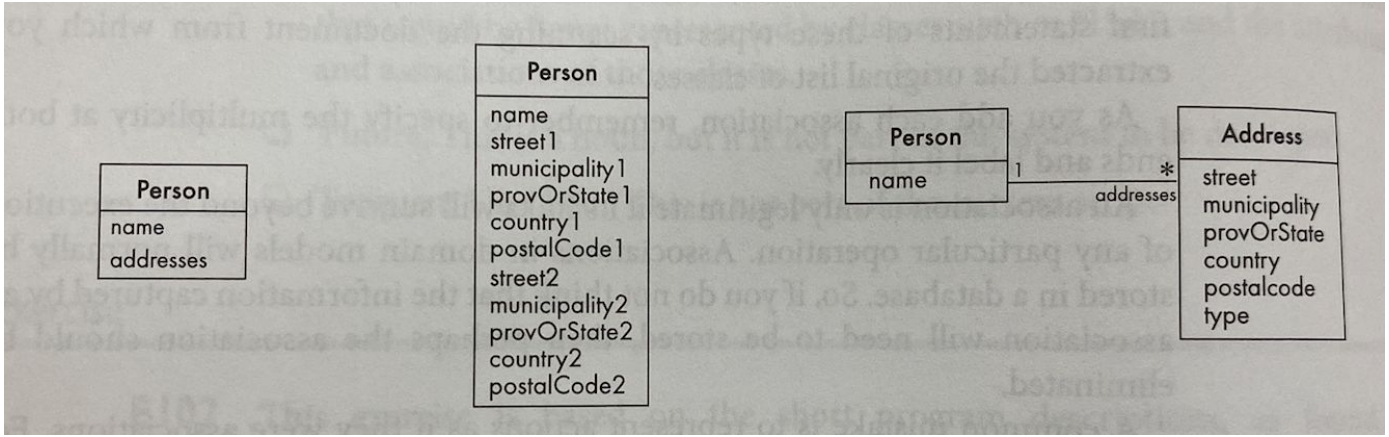
```
public final class Singleton {
    private static final Singleton INSTANCE = new Singleton();

    private Singleton() {}

    public static Singleton getInstance() {
        return INSTANCE;
    }
}
```

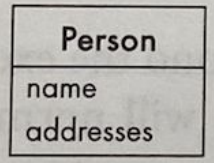
We have a sequential file access and … file access

Explain Why do we need it?

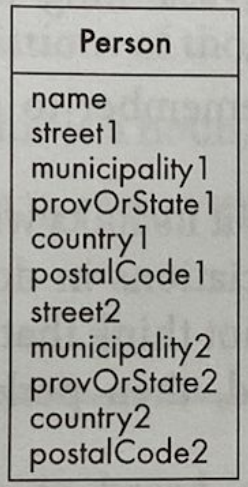


Make design choice and explain

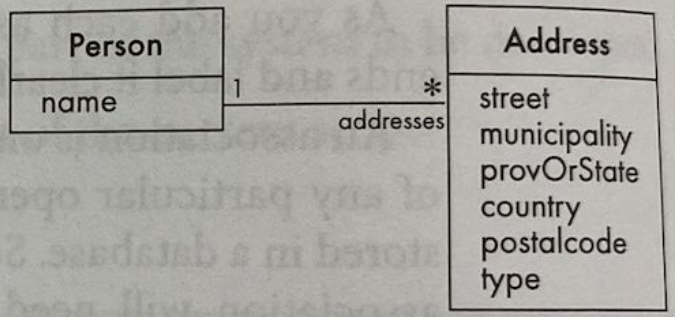




Bad, due to a plural attribute



Bad, due to too many attributes, and the inability to add more addresses



Good solution. The type indicates whether it is a home address, business address etc.

1. You can create object of the interface
2. You can declare interface as a variable
3. You can instantiate the interface

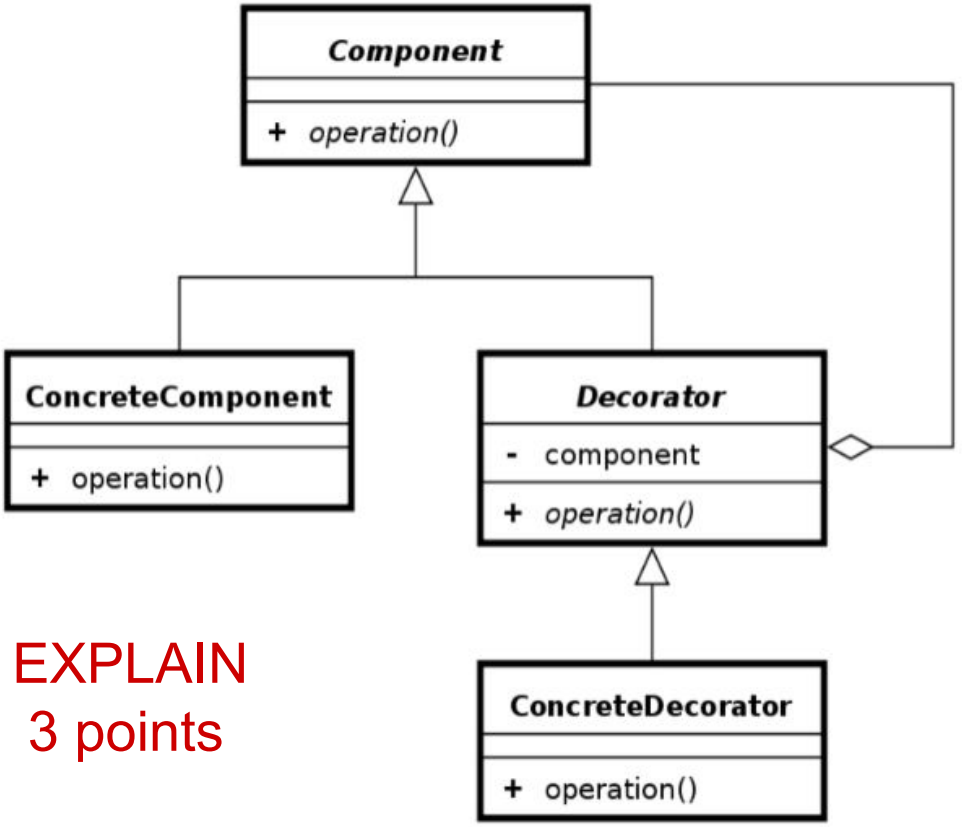
Which of these are true?

a. 1, 2
b. 3
c. 2
d. 1
e. 2,3
f. none

Give example of the interface and abstract class at the educational web site containing animations and describing how algorithms work

What is the output?

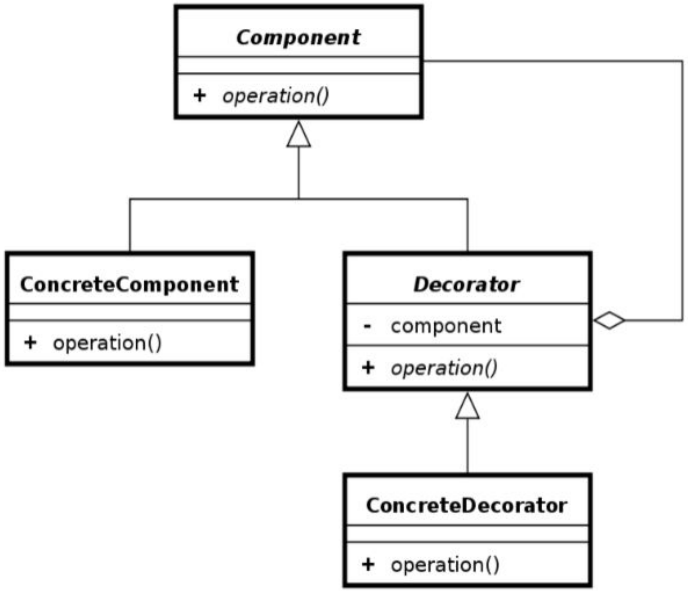
```
class Toy {
  Toy() {
    System.out.println("toy");
  } }
class Barbi extends Toy {
  Barbi() {
    System.out.println("barbi");
  } }
class OrderOfConstruction {
  public static void main(String args[]) {
    Barbi test = new Barbi();
   } }
```



EXPLAIN
3 points

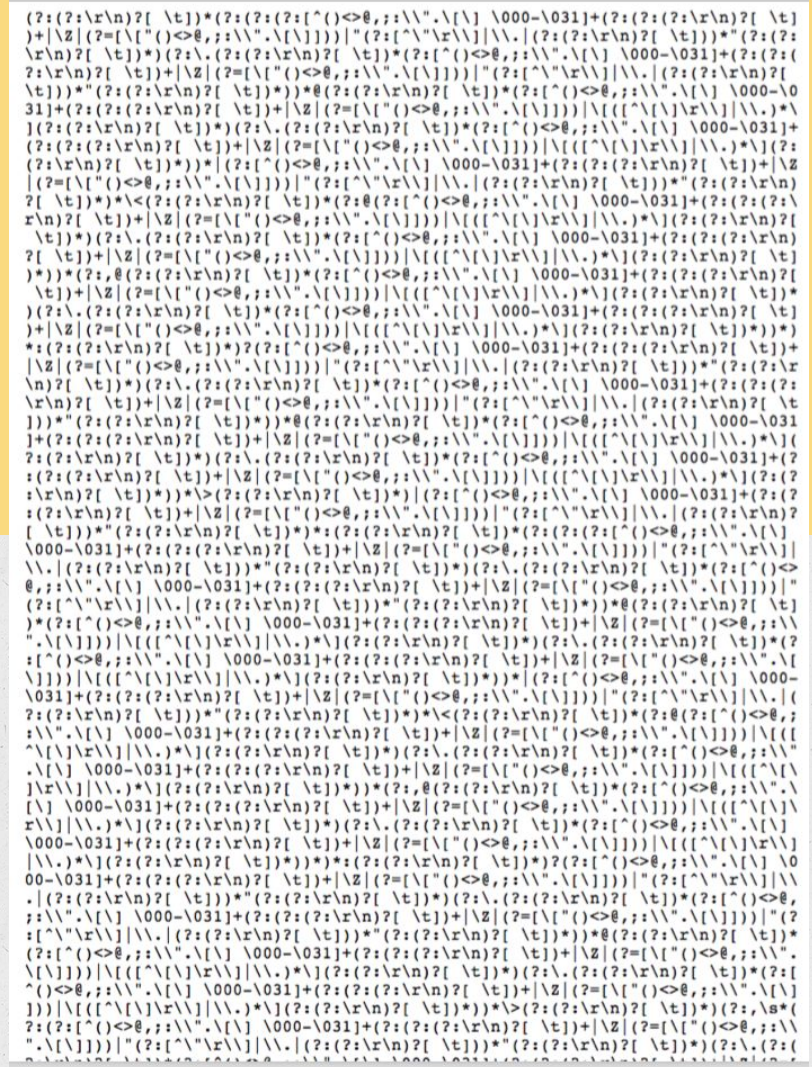
Decorator(Wrapper) Pattern

It is a design pattern that allows behavior to be added to an individual object, either statically or dynamically, without affecting the behavior of other objects from the same class



The decorator pattern can be used to extend (decorate) the functionality of a certain object statically, or in some cases at run-time, independently of other instances of the same class

What is Serialization? Which classes allow us to realize it?



. (2 points) Why an EQUALS() method MUST accept a parameter of type OBJECT? Provide valid reasons. Why we need a hashcode?

5 points

** The next level following after OOP is functional programming, where inputs and outputs can be functions. For example, in Scala (another functional languages you can try: f#, Haskell, Clojure, OCaml…) we can define the function that accepts a function as a parameter:

```
def fold(f: (Int)=>Unit, b:Int, c:Int):Unit= if(b<=c) {f(b); fold(f, b+1, c) }
```

So, this function accepts as parameters a function f that takes Int and returns nothing (Unit is like void), and two integers b and c. What it does is applies function f to b increasing b while b is <=c. Your task is to create *smth similar* in java – some method that can accept a function as a parameter.

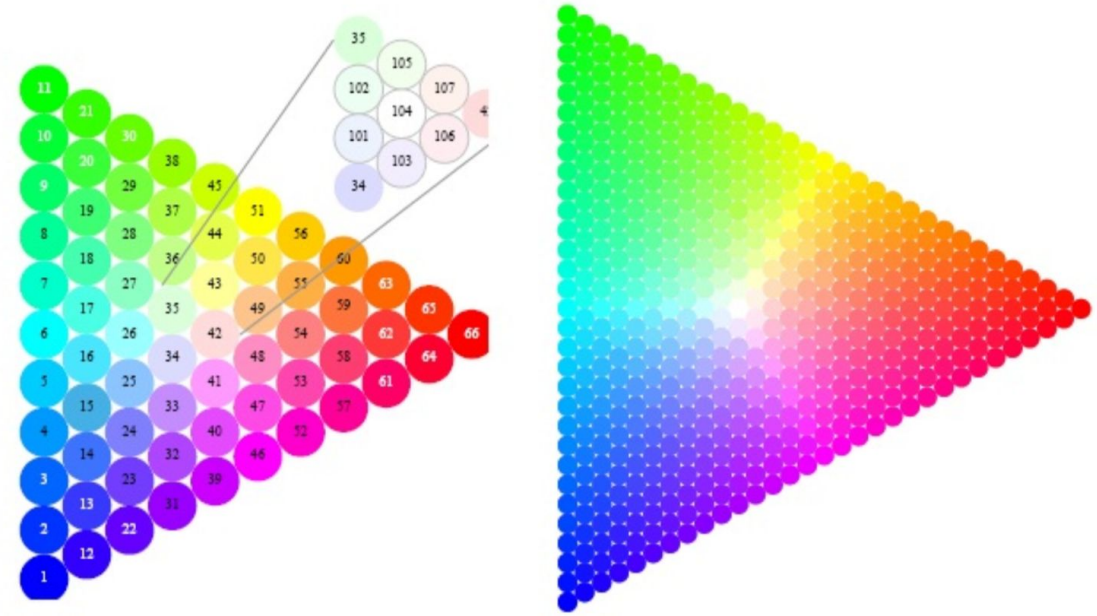
2 points

*** **Coupling** refers to the degree to which one class relies on knowledge of the internals of another class. Classes can be *tightly* coupled (high coupling) or *loosely* coupled (low coupling).

Cohesion refers to what the class (or module) can do. *Low* cohesion would mean that the class does a great variety of actions - it is broad, unfocused on what it should do. *High* cohesion means that the class is focused on what it should be doing, i.e. only methods relating to the intention of the class. Example of low cohesion: Book and Author in one class, or Car and Driver in one class.
Now, based on this and your background knowledge, what do you think a desirable Object-oriented system design must be? (Write several sentences).

Idea - 1 point, pseudocode - 3 points

Write a program that takes 3 input colors from the user and displays such triangle. Corners of the triangle are 3 input colors. Each such circle must be numbered. At a side, you should display a list of numbers and the corresponding (r,g,b), (h,s,i), (c,m,y,k), (l, a, b) values. You can find the corresponding formulas in the internet.



Thank you for attention!