**OBJECT ORIENTED PROGRAMMING**

**Assignment 2**

1. 51. Give an example among tight coupling and loose coupling.

<https://www.geeksforgeeks.org/coupling-in-java/#:~:text=Example%20%3A%20If%20you%20change%20your,then%20you%20have%20tight%20coupling>.

<https://www.upgrad.com/blog/loose-coupling-vs-tight-coupling-in-java/>

1. 52. Write in brief abstract class.

Ans. Following are some important observations about abstract classes in Java.

* An instance of an abstract class can not be created.
* Constructors are allowed.
* We can have an abstract class with or without any abstract method.
* Abstract classes can not have final methods because when you make a method final you can not override it but the abstract methods are meant for overriding.
* We are not allowed to create object(instantiate) for any abstract class.
* We can define static methods in an abstract class

1. 53. Define the Benefits of oops over pop?

Ans.

* OOPs makes development and maintenance easier where as in Procedure-oriented programming language it is not easy to manage if code grows as project size grows.
* OOPs provides data hiding whereas in Procedure-oriented programming language a global data can be accessed from anywhere.
* OOPs provides ability to simulate real-world event much more effectively. We can provide the solution of real word problem if we are using the Object-Oriented Programming language.

**REAL LIFE EXAMPLE**:

Imagine you clean up and put all your stuff into the furnace room. Great - you know where it all is. The old stereo is behind the Christmas decorations. You know that to to get to the hi fi, you have first move the shoes, then the golf clubs, the Christmas stuff, bingo - your'e there .. no sweat right ?

Now let's pretend you go back 2 years later ... not easy is it ? You can't remember where anything is, or how to get to it, and everything is strewn all over the floor anyway. That is procedural programming.

Now let's put everything into neat boxes, put labels on boxes. In fact we could even organise the shelves - that's object orientated programming. You come back 2 years later .. no problem.

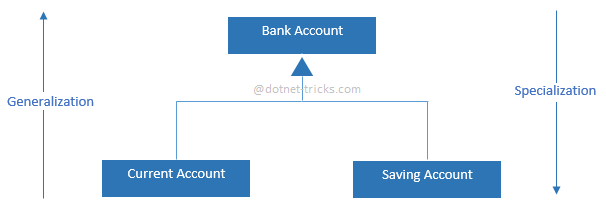
"objects" are just boxes, where you put things in. Those things could be "variables" like a name or an address - they don't do anything or "methods" which can do stuff like add 2 numbers or send an email.

1. 54. Explain Generalization and Specialization?

* The process of extracting common characteristics from two or more classes and combining them into a generalized superclass is called Generalization. The common characteristics can be attributes or methods. Generalization is represented by a triangle followed by a line.



* Specialization is the reverse process of Generalization means creating new subclasses from an existing class.
* Let’s take an example of Bank Account; A Bank Account is of two types – Current Account and Saving Account. Current Account and Saving Account inherits the common/ generalized properties like Account Number, Account Balance, etc. from a Bank Account and also have their own specialized properties like interest rate, etc.



The following are the major differences between generalization and specialization:

1. **Structure:** Generalization is a bottom-up approach in that it begins with lower-level, or narrower, entity types and broadens them to higher-level types. Specialization, in contrast, is a top-down approach since it creates lower-level entity types out of higher-level ones.
2. **Basis:** Common attributes between different entities form the basis for generalization. For specialization, it's uncommon attributes.
3. **Number:** Generalization involves multiple entities and combines them into a generalized entity. Specialization involves a single entity broken down into multiple sub-entities.
4. **Size:** Generalization reduces the schema of the data by unifying components. Specialization expands the schema by multiplying the components.
5. **Inheritance:** Inheritance is the capability of a lower-level entity to retain attributes from higher-level associations. Inheritance is possible with specialization but not generalization.
6. **Use:** Generalization allows users to visualize bigger-picture patterns. Specialization allows users to narrow their search.
7. 55. Write in brief Association, Aggregation and Composition?

**Association**

Association refers to the relationship between multiple objects. It refers to how objects are related to each other and how they are using each other's functionality. Composition and aggregation are two types of association.

**Composition**

The composition is the strong type of association. An association is said to composition if an Object owns another object and another object cannot exist without the owner object. Consider the case of Human having a heart. Here Human object contains the heart and heart cannot exist without Human.

**Aggregation**

Aggregation is a weak association. An association is said to be aggregation if both Objects can exist independently. For example, a Team object and a Player object. The team contains multiple players but a player can exist without a team.

Example Code Link: <https://www.tutorialspoint.com/Association-Composition-and-Aggregation-in-Java>

1. 56. Write in brief Object Composition vs. Inheritance.

* The composition is a design technique in which your class can have an instance of another class as a field of your class. Inheritance is a mechanism under which one object can acquire the properties and behavior of the parent object by extending a class.
* Composition and Inheritance both provide code reusability by relating class.  We can also get the functionality of inheritance when you use composition.
* Below are the differences.

| **Sr. No.** | **Key** | **Inheritance** | **Composition** |
| --- | --- | --- | --- |
| 1 | Basic | Inheritance is an "is-a" relationship | Composition is a "has-a". Relationship |
| 2 | Code Reuse | In Inheritance, a class lass can extend only one interface, therefore, you can reuse your code only in one class only | We can reuse code in multiple class |
| 3 | Scope | Inheritance provides its features at compile time | Composition is easily achieved at runtime |
| 4 | Final | We can’t reuse code from the final class | It allows code reuse even from final classes |
| 5 | Methods | It exposes both public and protected method of the parent class | It doesn’t expose. They interact using public interface. |

1. 57. Explain cohesion?
2. 58. Explain “black-box-reuse” and “white-box-reuse”?
3. 59. Explain “this”
4. 60. Write in brief static member and member function.
5. 61. How will you relate unrelated classes or how will you achieve polymorphism without using base class?
6. 62. Explain Diamond problem?
7. 63. Explain the solution for diamond problem?
8. 64. Explain the need of abstract class?
9. 65. Why can’t we instantiate abstract class?
10. 66. Can abstract class have constructors?
11. 67. How many instances can be created for an abstract class?
12. 68. Which keyword can be used for overloading?

Ans.There is no such keyword for Overloading in java.

1. 69. Explain the default access specifiers in a class definition?
2. 70. Define all the operators that cannot be overloaded?

Ans. In java no such provision is there. Its regarding C++. Ignore question.

<https://www.tutorialspoint.com/operators-that-cannot-be-overloaded-in-cplusplus#:~:text=For%20an%20example%20the%20sizeof,Arjun%20Thakur>

1. 71. Explain the difference among structure and a class?
2. 72. Explain the default access modifier in a class?
3. 73. Can you list out the different types of constructors?
4. 74. Explain a friend function?
5. 75. Explain a ternary operator?
6. 76. Do We Require Parameter For Constructors?
7. 77. Explain Sealed Modifiers?
8. 78. Explain The Difference Among New And Override?
9. 79. How Can We Call The Base Method Without Creating An Instance?
10. 80. Define The Various Types Of Constructors?
11. 81. Define Manipulators?
12. 82. Can you give some examples of tokens?