1. Explain polymorphism and why it is important.

-the ability of an object to take many forms and it allows us to perform the same action in many different ways

-it’s important because it allows me to map a method call to the concrete type, based on runtime type of the object

1. Explain the open close principle and give an example.

-according to the open-closed principle, software should be designed so that it is open for extension, but closed for modification.

-excellent example is that if we work with a framework (like Spring), the only way to extend functionality is by adding new classes since we do not have the option to modify the framework code directly. This means that Spring's framework code adheres 100% to the open-closed principle.

1. Explain early binding and when it is possible.

-binding during compile time, also called as static binding of the method call

-binding of all the static, private and final methods is done at compile time.

-it’s possible when the compiler maps the method at compile time

-e.g; method overloading

1. Explain late binding and why it is needed.

-binding during runtime, also called as dynamic binding

-It is needed when we want to make binding of a method according to the runtime type of the object where polymorphism is implemented

-e.g; method overriding

1. Explain programming to an interface and what are the advantages of doing so.

-a technique to write classes based on an interface

-It involves creating an interface first, defining its methods and then creating the actual class with the implementation

-when we program to an interface, we depend on the interface, rather than on the concrete type

-Advantages

-it gives open-closed principle

-it promotes encapsulate change and extensibility

-it makes the class much more flexible and easier for maintenance

1. Explain Factory design pattern and why is it important

-excellent example of programming to an interface

-a way of encapsulating the logic of object creation which is typically done in constructor

-it is important because it provides an approach to code for interface rather than implementation.

1. List at three advantages of using a Factory method over using the constructor

-Factory methods have a name – easier to understand what is being requested, and to distinguish between different kinds of invocations on an object

-Can control access to instances.

-Solves the problem that a class can have only one constructor with a given signature

-Unlike constructors, factory methods are not required to create a new instance every time they are invoked.

-Unlike constructors, factory methods can return a subtype of the requested type, or an implementation of an interface type only if it is not null.

1. Explain Template Method design pattern and how it is useful

-Sometimes a class at the top of an inheritance hierarchy needs to carry out a sequence of tasks, some of which need to be implemented by subclasses. This situation is an example of the Template Method Design Pattern.

-It is useful for avoiding code duplication and letting subclasses implement varying behavior through method overriding.

1. Explain Listener design pattern and give an example of its application

-a behavioral design pattern that allows some objects to notify other objects about changes in their state.

-we handle events by using listener design pattern

-It allows us dynamically to add or remove an listener to give an event

obj.addActionListener(new ActionListener( ) {

public void actionPerformed(ActionEvent e) {

}

});

-above sample is gonna notify when an action happened or occurred

1. Explain the Façade design pattern and give an example of how it is useful for information hiding in subsystem design

-The facade pattern is appropriate when we have a complex system that we want to expose to clients in a simplified way.

-Its purpose is to hide internal complexity behind a single interface that appears simple from the outside.

-It does not restrict access to sub-systems

-Example:

calling dataSource.getConnection() to get the connection but internally a lot of things happen such as loading the driver, creating connection or fetching connection from pool, update stats and then return the connection reference to caller method

1. Explain The Singleton design pattern and show how you implement it.

-This pattern involves a single class which is responsible for creating an object while making sure that only a single object gets created.

-This class provides a way to access its only object which can be accessed directly without need to instantiate the object of the class.

-We can implement it as the following,

public class MySingleton{

private static MySingleton instance = new MySingleton();

private MySingleton(){}

public static MySingleton getInstance(){

return instance;

}

}