OOPS Assignment Solution

Question 1: What is Inheritance in Java?

Answer 1: Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOPs (Object Oriented programming system).

- The idea behind inheritance in Java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of the parent class. Moreover, you can add new methods and fields in your current class also.
- Inheritance represents the IS-A relationship which is also known as a parent-child relationship.
- Terms used in Inheritance
    - Class: A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
    - Sub Class/Child Class: Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
    - Super Class/Parent Class: Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
    - Reusability: As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

Question 2: What is superclass and subclass?

Answer 2: **Sub Class/Child Class**: Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.

- **Super Class/Parent Class**: Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.

Question 3: How is inheritance implement/ achieved in java?

Answer 3: Inheritance in Java is implemented using the extends keyword. The extends keyword indicates that the child class inherits from the parent class. The child class then gains access to all of the properties and methods of the parent class, which it can either use directly or override.

Question 4: What is polymorphism?

Answer 4: Polymorphism in Java is a concept by which we can perform a single action in different ways. Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So, polymorphism means many forms.

- There are two types of polymorphism in Java: compile-time polymorphism and runtime polymorphism. We can perform polymorphism in java by method overloading and method overriding.

Question 5: Difference between method overloading and overriding?

Answer 5:

Method Overloading	Method Overriding
Method overloading is a compile-time polymorphism.	Method overriding is a run-time polymorphism.
Method overloading helps to increase the readability of the program.	Method overriding is used to grant the specific implementation of the method which is already provided by its parent class or superclass.
Method overloading may or may not require inheritance.	Method overriding always needs inheritance.
In method overloading, methods must have the same name and different signatures.	In method overriding, methods must have the same name and same signature.

Question 6: What is an abstraction in java explained with example?

Answer 6: In Java, abstraction is achieved by interfaces and abstract classes. We can achieve 100% abstraction using interfaces. Data Abstraction may also be defined as the process of identifying only the required characteristics of an object ignoring the irrelevant details. The properties and behaviors of an object differentiate it from other objects of similar type and also help in classifying/grouping the objects.
Example:

```
abstract class Bike
        {
          abstract void run();
        }
class Honda4 extends Bike
{
        void run()
        {
                System.out.println("running safely");
        }
public static void main(String args[])
```

```
{
        Bike obj = new Honda4();
        obj.run();
}
}
```

Question 7: what is the difference between an abstract method and final method in java? Explain with an example

Answer 7:

Feature	Final Method	Abstract Method
Definition	A method that is marked as final cannot be overridden in subclasses.	A method that is not implemented in the parent class and must be overridden in subclasses.
Use	A final method is used to prevent a method from being overridden in subclasses.	An abstract method is used to provide a common interface for subclasses to implement.

Example	final class ImmutableClass {
	  private final String name;
	
	  public ImmutableClass(String name) {
	  this.name = name;
	  }
	
	  public String getName() {
	  return name;
	  }
	}
	
	public class Main {
	  public static void main(String[] args) {
	  ImmutableClass immutableObject = new ImmutableClass("Hello");
		abstract class Bike
		  {
		  abstract void run();
		  }
		  class Honda4 extends Bike
		  {
		  void run()
		  {
		  System.out.println("running safely");
		  }
		  public static void main(String args[])
		  {
		  Bike obj = new Honda4();
		  obj.run();
		  }

	System.out.println(immutableObject.getName()); } }	}

Example:

Question 8: What is final class in java?

Answer 8: A class which is declared with the “Final” keyword is known as the final class. The final keyword is used to finalize the implementations of the classes, the methods and the variables used in this class. The main purpose of using a final class is to prevent the class from being inherited (i.e.) if a class is marked as final, then no other class can inherit any properties or methods from the final class. If the final class is extended, Java gives a compile-time error

Question 9: Difference between abstraction and encapsulation.

Answer 9:

Abstraction	Encapsulation
Abstraction is the process or method of gaining the information.	While encapsulation is the process or method to contain the information into a single unit and providing this single unit to the user.
Main feature: reduce complexity, promote maintainability, and also provide clear separation between the interface and its concrete implementation.	Main feature: Data hiding. It is a common practice to add data hiding in any real-world product to protect it from external world. In OOPs, this is done through specific access modifiers.
In abstraction, problems are solved at the design or interface level.	While in encapsulation, problems are solved at the implementation level.
Abstraction is the method of hiding the unwanted information.	Whereas encapsulation is a method to hide the data in a single entity or unit along with

	a method to protect information from outside.
We can implement abstraction using abstract class and interfaces.	Whereas encapsulation can be implemented using by access modifier i.e. private, protected and public.

Question 10: Difference between Runtime and Compile time polymorphism with an example.

Answer 10:

Compile Time Polymorphism	**Run time Polymorphism**
In Compile time Polymorphism, the call is resolved by the compiler.	In Run time Polymorphism, the call is not resolved by the compiler.
It is also known as Static binding, Early binding and overloading as well.	It is also known as Dynamic binding, Late binding and overriding as well.
Method overloading is the compile-time polymorphism where more than one method shares the same name with different parameters or signature and different return type.	Method overriding is the runtime polymorphism having the same method with same parameters or signature but associated with compared, different classes.
It is achieved by function overloading and operator overloading.	It is achieved by virtual functions and pointers.
It provides fast execution because the method that needs to be executed is known early at the compile time.	It provides slow execution as compare to early binding because the method that needs to be executed is known at the runtime.