1. What are the four pillars of Object-Oriented Programming? Explain each pillar.

One of the four pillars of Object-Oriented Programming is Abstraction. This includes only showing basic information to user and hiding complexity. Another pillar is Encapsulation. This is basically binding data and functions together. A third pillar is Inheritance. This includes the ability to acquire functionality/features from a related class; child/parent. And finally, Polymorphism. Which is when an object/function can have different behaviors depending on the context.

https://docs.oracle.com/javase/tutorial/java/concepts/index.html

https://www.codingninjas.com/codestudio/library/four-pillars-of-oops#:~:text=The%20definition%20of%20data%20structures,make%20up%20these%20four%20pillars.

2. What is the relationship between a Class and an Object?

The relationship between Class and an Object is that Class is like a blueprint and an Object is an example of something made from that blueprint.

3. What are the differences between checked and unchecked exceptions?

The main difference between checked and unchecked exceptions is that checked exceptions are checked when the code is compiled whereas unchecked exceptions are brought to attention after the code has ran. Unchecked exceptions relate more to the logic of the code rather than the syntax.

https://www.baeldung.com/java-checked-unchecked-exceptions

https://www.geeksforgeeks.org/checked-vs-unchecked-exceptions-in-java/

4. What are the differences between abstract classes and interfaces? When should you use one over the other?

There are a couple differences between abstract classes and interfaces, one of which is that an interface can only contain abstract methods but an abstract class can also contain non-abstract methods. Another difference is that you can only implement one abstract class with a class but you can implement multiple interfaces. If having more control over access modifiers is important than using an abstract class would be beneficial.

https://www.geeksforgeeks.org/difference-between-abstract-class-and-interface-in-java/

https://docs.oracle.com/javase/tutorial/java/IandI/abstract.html

5. What is unit testing and why is it important?

Unit testing is checking the functionality of your code by testing it in smaller sections. It is very important when writing code because unit testing allows you to make sure everything is doing what it is supposed to do. This also makes it easier to troubleshoot if there is an issue.

6. What is your favorite thing you learned this week?

I really enjoyed everything this week, Classes/Objects, Abstract Classes/Interfaces, Exceptions, but the Four Pillars of Object-Oriented Programming were very interesting because they helped me gain a larger perspective when thinking about and writing code.