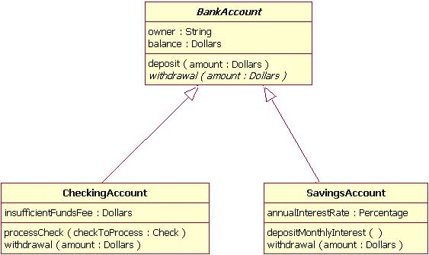
Java 8-2: Inheritance, Interfaces Related exercises

1. The slides showed a class diagram of BankAccount with two subclasses CheckingAccount and SavingsAccount.



1. Should BankAccount be a concrete class, abstract class or interface? Does it matter?
2. Do the subclasses know about member variables owner and balance in BankAccount?
3. What overloaded constructors, if any, would you want to use in the subclasses?
4. If you model BankAccount as an abstract class, can it have constructor(s)?
5. Will this line of code compile? BankAccount ba = new BankAccount(); if BankAccount is an interface or abstract class?
6. Will these lines compile? BankAccount ba = new CheckingAccount();

BankAccount ba = new SavingsAccount();

1. What principle is part(f) demonstrating?
2. Assuming you can use BankAccount ba = new CheckingAccount(); what happens if we use code ba.processCheck(checkToProcess) assuming checkToProcess is a valid instance?
3. Now create a Java project using the information from the class diagram. Try to use good programming practices with respect to design, naming and other conventions. Create at least one CheckingAccount and one SavingsAccount. Use all the methods and variables for the classes. You can use numeric types like int, double (or Integer, Double) for the variables, parameters instead of Dollars, Percentage if you want. You can be creative in your project. Create a BankDriver class with a main method to check your code. Then add methods like equals() and hashCode(), toString(), any other methods you think are appropriate, and a method to compare account type objects. What would you use here? Can you implement or extend the interface you need in BankAccount or do you have to do the same in the subclasses?
4. Do you think there are two types of polymorphism – compile time and runtime? Explain.
5. Can you overload static methods?
6. For an inheritance type hierarchy and relationship should the top most “class” be an abstract class or interface? Explain.
7. What does it mean to favor composition over inheritance? Aren’t these different concepts or does this imply one can be substituted for the other? Give an example of why this can be important and some of the advantages and disadvantages of inheritance.
8. If a marker interface like Cloneable, Serializable, Remote have methods or constants what use are they?
9. What is a default method and how does it work? Can we have more than one default method in an interface?
   1. **Default method is an option for interfaces in order to avoid requiring some methods to be abstract only meaning they have to be implemented in the class using the interface, but this way, we can have a default method in the interface itself.**
   2. **Yes, we can have more than one default method.**
   3. **Default methods were introduced so we don’t necessarily have to treat every method as abstract and we can have method implementations in our interfaces.**
10. Explain at least one reason for introducing default methods in Java 8. What is a default method and how does it work? Can we have more than one default method in an interface?
    1. **// duplicate question**
11. What is the purpose of a static method in an interface?
    1. **It can’t be overridden in the implementation class.**
12. Is it good practice to have an interface that has only constants?
    1. **No, we could use enum which is more efficient.**