Career Services Assignment 3 – Java Flash Cards

Points possible: 50

Category	Criteria	% of Grade
Completeness	All requirements of the assignment are complete.	100

Instructions: Research common Java interview questions online and create 20 flash cards from the information you find. Study your flash cards regularly to better prepare for interviews. Fill out the table below with the information you put on each of your flash cards.

Front of Card	Back of Card

1. What are the principle concepts of OOPS?	There are four principle concepts upon which object-oriented design and programming rest. They are: • Abstraction • Polymorphism • Inheritance • Encapsulation. (i.e. easily remembered as A-PIE).
2. What is Abstraction?	Abstraction refers to the act of representing essential features without including the background details or explanations.
3. What is Encapsulation?	Encapsulation is a technique used for hiding the properties and behaviors of an object and allowing outside access only as appropriate. It prevents other objects from directly altering or accessing the properties or methods of the encapsulated object.

4. What is the difference between Abstraction and Encapsulation?	Abstraction focuses on the outside view of an object (i.e. the interface) Encapsulation (information hiding) prevents clients from seeing its inside view, where the behavior of the abstraction is implemented. ▪ Abstraction solves the problem on the design side while Encapsulation is the Implementation. ▪ Encapsulation is the deliverables of Abstraction. Encapsulation barely talks about grouping up your abstraction to suit the developer's needs.
5. What is Inheritance?	Inheritance is the process by which objects of one class acquire the properties of objects of another class. A class that is inherited is called a superclass. The class that does the inheriting is called a subclass. Inheritance is done by using the keyword extends. The two most common reasons to use inheritance are: • To promote code reuse • To use polymorphism.
6. What is Polymorphism?	Polymorphism is briefly described as “one interface, many implementations.” Polymorphism usage to something in different contexts – specifically, to allow an entity such as a variable, a function, or an object to have more than one form.

7. How does Java implement polymorphism?	Inheritance, Overloading, and Overriding are used to achieve Polymorphism in Java. Polymorphism manifests itself in Java in the form of multiple methods having the same name. In some cases, multiple methods have the same name, but different formal argument lists (overloaded methods). In other cases, multiple methods have the same name, same return type, and the same formal argument list (overridden methods).
8. Explain the different forms of Polymorphism.	There are two types of polymorphism, one is Compile time polymorphism and the other is run time polymorphism. Compile-time polymorphism is method overloading. Runtime time polymorphism is done using inheritance and interface. **Note**: From a practical programming viewpoint, polymorphism manifests itself in three distinct forms in Java: • Method overloading • Method overriding through inheritance • Method overriding through the Java interface.

9. What is runtime polymorphism or dynamic method dispatch?	In Java, runtime polymorphism or dynamic method dispatch is a process in which a call to an overridden method is resolved at runtime rather than at compile time. In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.
10. What is Dynamic Binding?	Binding refers to the linking of a procedure call to the code to be executed in response to the call. Dynamic binding (also known as late binding) means that the code associated with a given procedure call is not known until the time of the call at run-time. It is associated with polymorphism and inheritance.

11. What is method overloading?	Method Overloading means having two or more methods with the same name in the same class with different arguments. The benefit of method overloading is that it allows you to implement methods that support the same semantic operation but differ by argument number or type. **Important Note:** • Overloaded methods MUST change the argument list. • Overloaded methods CAN change the return type. • Overloaded methods CAN change the access modifier. • Overloaded methods CAN declare new or broader checked exceptions. • A method can be overloaded in the same class or in a subclass.

12. What is method overriding?	Method overriding occurs when a subclass declares a method that has the same type of arguments as a method declared by one of its superclasses. The key benefit of overriding is the ability to define behavior that's specific to a particular subclass type. **Note**: The overriding method cannot have a more restrictive access modifier than the method being overridden (Ex: You can't override a method marked public and make it protected). You cannot override a method marked final. You cannot override a method marked Static.

13. What is the difference between method overloading and method overriding?	**Overloaded Method:** • Arguments – Must Change • Return type: Can Change • Exceptions – Can Change • Access – Can Change • Invocation – Reference type determines which overloaded version is selected. Happens at compile time. **Overridden Method:** • Arguments – Must not Change • Return type: Can't change except for covariant returns • Exceptions – Can reduce or eliminate. Must not throw new or broader checked exceptions • Access – Must not make more restrictive (can be less restrictive)

	• Invocation – Object type determines which method is selected. Happens at runtime.

14. Can overloaded methods be overridden too?	Yes, derived classes still can override the overloaded methods. Polymorphism can still happen. The compiler will not be binding the method calls since it is overloaded, because it might be overridden now or in the future.
15. Is it possible to override the main method?	No, because the main is a Static method. A Static method can't be overridden in Java.

16. How to invoke a superclass version of an Overridden method?	To invoke a superclass method that has been overridden in a subclass, you must either call the method directly through a superclass instance or use the super prefix in the subclass itself. From the point of view of the subclass, the super prefix provides an explicit reference to the superclass's implementation of the method.
	
	`// From subclass`
	`super.overriddenMethod();`

17. What is Super?	Super is a keyword that is used to access the method or member variables from the superclass. If a method hides one of the member variables in its superclass, the method can refer to the hidden variable through the use of the super keyword. In the same way, if a method overrides one of the methods in its superclass, the method can invoke the overridden method through the use of the super keyword. Note: You can only go back to one level. In the constructor, if you use super(), it must be the very first code, and you cannot access any of these. xxx variables or methods to compute its parameters.

18. How do you prevent a method from being overridden?	To prevent a specific method from being overridden in a subclass, use the final modifier on the method declaration, which means “this is the final implementation of this method”, the end of its inheritance hierarchy.
	
	`public final void exampleMethod() {`
	`// Method statements`
	`}`

19. What is an Interface?	An interface is a description of a set of methods that conform to implementing classes must-haves. **Note:** • You can’t mark an interface as final. • Interface variables must be Static. • An Interface cannot extend anything but another interface.

20. Does not overriding hashCode() method has any performance implication?	This is a good question and opens to all, as per my knowledge, a poor hash code function will result in the frequent collision in HashMap which eventually increases the time for adding an object into Hash Map. From Java 8 onwards though collision will not impact performance as much as it does in earlier versions because after a threshold the linked list will be replaced by a binary tree, which will give you **O(logN)** performance in the worst case as compared to O(n) of a linked list.