Homework 7

Q1: There are two of class instances in the same file. Any **public** classes must be separate .java file. Movie class definition must be removed. In addition, there is an inheritance example in this code but the **super** constructor method missing. Therefore, this is another line to be removed. Finally, there are two lines needs to remove.

Answer is option C

Q2: The **abstract** method can declared with no body. Its written in the method heading. An interface can have methods and variables like a class but these are abstract. Interface methods cannot be declared as private, protected or transient, The interface methods are abstract and public by default in java.

Answer is option D

Q3: The following application, there is a two method has same name and both of them are has no **argument**. If both of the methods has different parameter types, so they have different method signature. Moreover, this is possible and named with **overloading**. If there are no parameters, the method names should be different. In addition, the second playMusic method should return a integer value but it's did not. Thus, the code not compiled.

Answer is option C

Q4: **Inheritance** allows you to define a very general class and then later define more specialized classes that add some new methods or attributes to the existing general class definition. Objects use the commonly used attributes and methods. Primitive types does not have methods İts defined a data filed. Inheritance allows subclasses to access the parent class's public attributes and methods. Inheritance provides the repetition of defining methods and data fileds. So this is leads to simpler code. But there should be an complex data structure on the design.

Answer is option A

Q5: An **interface** is a kind of reference type and we can use it to declare variables and method parameters as having an interface type. It can be used in heading of the method name correctly. The Class violates the java naming convention and defining a class in the class structure not allowed.

Answer is option A

Q6: An **interface** is just the declaration of methods an object. It dos not contain the implementation. It a collection of abstract methods. İnterface indicates the method/s has to implemented it is useful for working with different teams in same project. It does not prevent any others work with using interface.

Answer is option B

Q7: The Electric car should be another class file because there are two of class declaration in a file. Therefore this code not complied.

Answer is option D

Q8: **Multiple inheritance** defined as a class can inherit the properties of more than one single parent class. In other words, a class extends from more than one class. Java not allowed this feature in direct manner. It can be surpassed with using of interfaces. When the abstract class is used for creating multiple inheritance, the compiler does not know which method to execute. This situation occurs the diamond shaped class diagram. This named as Diamond problem in java.

Answer is option D

Q9: To override a method means that child class provide a specific implementation of a method it defined in parent classes. When a method use the same name, same parameters and same return type its called the override the method in super class. To achieve this final keyword must be removed first. Then void type does not compatible with the Object type. The void keyword also deleted. The access modifier of overriding method can more accessible than then original method. A protected instance method in parent class can be made public modifier, not private and package-private. So this is last change for compiling.

Answer is option C

Q10: Overriding a method means that child class provide a specific implementation of a method it defined in parent classes. When a method use the same **name**, same **parameters** and same **return type** it's called the override the method in super class. The access modifier of overriding method can more **accessible** than then original method. The return type also become compatible with original method.

Another content of override is **exception handling**. There are two rules when overriding methods with exception handling. If a super class overridden method does **not** throw an exception subclass overriding method only throws the unchecked exception. Other rule is the If a super class overridden method throws an exception, subclass overriding method can only throw same subclass exception. The subclass may not throwing any exception.

Answer is option C

Q11: In this code there is an **final** keyword to remove. This violates the override method. And then the compile gives another error that the Laptop class must be defined another java file. The code is not compiled and not throws an exception.

Answer is option C

Q12: The code compiles properly. Then this has an object that it has the HighSchool type and the override in this class method running.
Answer is option A

Q13: Initiate objects of an interface may be implemented by classes that where is another package. Thus, the interface can take the **public** modifier. Interfaces include constants and its data members. This means the data members can declared as final and final constants are declared with **static** keyword to keep one instance of data members.

Answer is option B

Q14: **Default methods** (aka. Defender methods) which allow the interfaces to add new methods without affect the implementing classes at the moment. There is not any compilation error. Then the class override the walk method in main method. It returns the sprinting word.

Answer is option C

Q15: An **interface** can extend multiple interfaces.

A **class** can implement more than one interface at a time.

A **class** can extend only one **class**, but implement many **interfaces**.

An **interface** can extend another interfaces, in similar way **class** can extend another **class**.

Answer is option B

Q16: The code is not compiled because height data field represented as private and getHeight() method is not used.

Answer is option D

Q17: Java interface can have **default** and **static** method with Java 8. **Concrete** method that have some code in it. An abstract class can have abstract an non-abstract methods. These methods must be declared as **abstract** keyword. In interface side there is only **abstract** methods are allowed.

Answer is option D

Q18: This code does not compile at line g3 on IsoscelesRightTriangle because this class declared as an abstract and. Abstract classes cannot be instantiated.

Answer is option C

Q19: The code does not compile anyway. When we fill the blank with **Integer**, control flow looks for play method in abstract class because of return type integer value and across with compatibility problem of return type. Other way filled with the **Short**, control flow looks for interface and we came across with the compatibility problem again. Number is malfunction element in this question

Answer is option D

Q20: A **class** implements an **interface**, while a class extends an **abstract** class.

Answer is option C

Q21: The Encyclopedia class extends a book class and gets the constructor of its parent's class. And then getMaterial() Method returns parent class's material variable with the **super** key. Finally the main prints the papayrus on the screen.

Answer is option A

Q22: If the unknownBunny is a Bunny type variable and it can use methods, variables as myBunny reference. unknownBunny may be an abstract, and it shows only the object not any implementation. And it may be different type than my Bunny and casting used for the point the Bunny object.

Answer is option B

Q23: A private abstract method is useless when you use it occurs the compiler error. If subclass in different package default accessibility for abstract method can have issues. When there is an inheritance with abstract method, subclasses see the members of superclass. To make it the abstract method have a protected access modifier.

Answer is option D

Q24: The code does not compile because Sphere an interface and it cannot be the superclass of Mars. A superclass must be class not an interface

Answer is option C

Q25: Objects in Java is likely to have multiple types. The reference to the object is not the object itself. The nature of the object being created is never going to change. The object may be accessed through references of different compatible types. We can upcast the the reference without an explicit cast to the parent class object. This is called a widening reference conversion. If the going down the type hierarchy or downcast we also call this narrowing reference conversion. This is always requires an explicit cast.

Answer is option B

Q26: An interfaces implicitly abstract. We don't need to use the **abstract** keyword.

Answer is option B

Q27: When the execution starts the block with start static keyword is placed. It prints one. Then there are another initialization block over there and prints three. After that the constructor is excuted and printed two. Then the flow enters the blueCar class prints the four and 5 sequantially.

Answer is option C

Q28: Overloaded an overridden methods always have the same method name. Overloaded methods have different representation of parameters and return type.

Answer is option C

Q29: The Ball class (represented as Q29 in code) takes five as an argument of super class. It's an also final variable. There is an soccerBall type object created by casting equipment. IT returns the five value.

Answer is option A

Q30: The overriding method has the same name, number and type of parameters, and return type as the method it overrides. Basically it's definition of method hiding in Java.If a subclass defines a class method with the same signature as a class method in the superclass, the method in the subclass hides the one in parent class. Accoring to this explanation. This text filled like:

A class that defines an instance variable with the same name as a variable in the parent class is referred to as hiding a variable, while a class that defines a static method with the same signature as a static method in a parent class is referred to as hiding a method.

Answer is option C

Q31: In java we cannot override the **static** method. Because method overriding is based on dynamic binding at runtime. **Static** keyword modifies the lifecycle of variable and method. If you specify static method or a variable loaded at the they created. These are volatile. Nonstatic variables and methods available during runtime.

Answer is option A

Q32: The code does not compile because Rotorcraft (as represented Q32 in the code) is not instantiated and there is an abstract integer type method fly. Using of the abstract method a class must be declared as an abstract keyword.

Answer is option D

Q33: A class may be assigned to an **superclass** reference variable automatically *(upcasting)* but requires an explicit cast when assigned to a **subclass** reference variable.*(downcasting)*

Answer is option B

Q34: A **concrete class** is the first non-abstract subclass that is required to implement all of the inherited abstract methods. Other options are all about abstraction. İnterface is also abstract by default.

Answer is option D

Q35: The code has three compile errors . One of them is abstract methods can not include a method body. Two is fly method in the Bird (represented as Q35) class, has lack of return statement. And Third one final class can not have any subclasses.

Answer is option D

Q36: Both abstract classes and interfaces contain abstract method. An abstract class can have concrete methods too.

Answer is option D

Q37: The code does not compile because there is an duplicate method talk() is written in interface SpeakDialogue and SingMonologue. Etiher of the methods should be override.

Answer is option C

Q38: A virtual method can be overridden by inheriting class by a method with the same signature to provide the polymorphic behavior. This means that every **non-static** method,

not include the **final** keyword and not a **private** method become a virtual method. The methods, which cannot inherited for polymorphic behavior is not a virtual method.

Answer is option A

Q39: An interface can extends another interface or interfaces. A class that implements interface must implements all the methods in interface.An interface implements another interface, while a class extends another class.

Answer is option B

Q40: The code compiled properly. Same name of variable names can not create the override situation. It returns the two.

Answer is option A

Q41: If we overriding a method overridden method must not be restrictive. Option C and option A (package private method) is eliminated. A final keyword provides the solidity a method. It can not modified. Option D is eliminated

Answer is option D

Q42: This code does not compiled it gives the method zoologist() error undefined for type Class Canine.

Answer is option D

Q43: Interfaces cannot be initialized. Since Java 8 **default** and **static** methods used in the interface. With Java 9 **private** methods used in in the interface.

Answer is option A

Q44: This code does not compile for Ballroom class not initialized.

Answer is option B

Q45: Overloaded methods have a different list of parameters, while overridden methods must have the exact same return type.

Answer is option A

Q46: If a parent class does not include a no-argument constructor placed by a default. A child calss must at least one constructor definition.

Answer is option B

Q47: The object type determines which attributes exist in memory, while the reference type determines which attributes are accessible by the caller

Answer is option D

Q48: