HOMEWORK 7
JAVA INHERITANCE

Qn	Ans	Explanation
1	C	Two lines of the program contain compilation errors. The name variable in the Cinema Class is private so it cannot be accessible from Movie class. Movie Class is defined as a constructor and it is missing an explicit super() statement so Cinema Class does not include a no argument constructor.
2	D	Public modifier can be applied to an abstract interface method.
3	C	The code does not compile because Song class contains two methods with the same method signature. There is a conflict in the code.
4	A	Inheritance allows objects to access commonly used attributes and methods.
5	A	Canine cannot be returned as an instance of Class because it does not inherit Class.
6	B	An interface allows easy integration once the other team's code is complete.
7	B	Output is "Driving electric car". The object created is an ElectricCar in the main() method, even if it is assigned to a Car reference. Due to polymorphism, the method from the ElectricCar will be invoked so output will be "Driving electric car".
8	D	Java allows multiple inheritance using interfaces.
9	C	There are three problems with this method override. First problem is that watch() method in the Television class is marked final. Second problem is that the return types void and Object are not covariant. Last problem is that access modifier in the child class must be the same or broader than in the parent class.
10	C	A checked exception thrown by the method in the parent class must be thrown by the method in the child class. Option C is incorrect.
11	C	The code does not compile because the process() method is declared final in the Computer class.
12	A	The output is "2". Due to polymorphism, the overridden version of the method in HighSchool is used, regardless of the reference type, and 2 is printed.
13	B	Static modifier can be applied to an interface method.
14	C	Having one class implement two interfaces that both define the same default method signature leads to a compiler error, unless the class overrides the default method. In this case, the Sprint class does override the walk() method correctly, therefore the code compiles without issue, and Option C is correct.
15	B	Option B is not true. An interface can implement another interface.
16	D	The code does not compile because super.height is not visible in the Rocket class, making Option D the correct answer. Even though the Rocket class defines a height value, the super keyword looks for an inherited version. Since there are none, the code does not compile. Note that super.getWeight() returns 3 from the variable in the parent class, as polymorphism and overriding does not apply to instance variables.
17	D	Excluding default and static methods, an abstract class can contain both abstract and concrete methods, while an interface contains only abstract methods.
18	C	The code does not compile, so Option D is incorrect. The IsoscelesRightTriangle class is abstract; therefore, it cannot be instantiated on line g3. Only concrete classes can be instantiated, so the code does not compile, and Option C is the correct answer. The rest of the lines of code compile without issue. A concrete class can extend an abstract class, and an abstract class can extend a concrete class. Also, note that the override of getDescription() has a widening access modifier, which is fine per the rules of overriding methods.
19	D	The play() method is overridden in Saxophone for both Horn and Woodwind, so the return type must be covariant with both. Unfortunately, the inherited methods must also be compatible with each other. Since Integer is not a subclass of Short, and vice versa, there is no subclass that can be used to fill in the blank that would allow the code to compile. In other words, the Saxophone class cannot compile regardless of its implementation of play(), making Option D the correct answer.
20	C	A class implements an interface, while a class extends an abstract class.
21	A	The code compiles and runs without issue, making Options C and D incorrect. Although super.material and this.material are poor choices in accessing static variables, they are

		permitted. Since super is used to access the variable in getMaterial(), the value papyrus is returned, making Option A the correct answer. Also, note that the constructor Book(String) is not used in the Encyclopedia class.
22	B	The data type of unknownBunny must be Bunny or a subclass of Bunny.
23	D	Protected modifiers can be applied to an abstract method.
24	D	The declaration of Sphere compiles without issue, so Option C is incorrect. The Mars class declaration is invalid because Mars cannot extend Sphere, an interface, nor can Mars implement Planet, a class. In other words, they are reversed. Since the code does not compile, Option D is the correct answer. Note that if Sphere and Planet were swapped in the Mars class definition, the code would compile and the output would be Mars, making Option A the correct answer.
25	B	A reference to a class can be assigned to a superclass reference without an explicit cast.
26	B	Abstract is not implicitly applied to all interface variables.
27	C	The output is "13245". The class is loaded first, with the static initialization block called and 1 is outputted first. When the BlueCar is created in the main() method, the superclass initialization happens first. The instance initialization blocks are executed before the constructor, so 32 is outputted next. Finally, the class is loaded with the instance initialization blocks again being called before the constructor, outputting 45.
28	C	Overloaded and overridden methods always have the same method name.
29	A	The output is "5". At runtime, the object is passed around and, due to polymorphism, can be read using any of those references since the underlying object is a SoccerBall. In other words, casting it to a different reference variable does not modify the object or cause it to lose its underlying SoccerBall information.
30	C	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.
31	B	The code does not compile due to line x2 because the override of getEqualSides() in Square is invalid.
32	C	The code does not compile because The Rotorcraft class includes an abstract method, but the class itself is not marked abstract.
33	B	A class may be assigned to a superclass reference variable automatically but requires an explicit cast when assigned to a subclass reference variable.
34	C	A concrete class is the first non-abstract subclass that is required to implement all of the inherited abstract methods.
35	D	The code contains three errors. The method fly() defined in CanFly is not marked static or default and defines an implementation, an empty {}, meaning it cannot be assumed to be abstract. The implementation of fly(int speed) in the Bird class also does not compile because of the signature. The Eagle class does not compile because it extends the Bird class.
36	B	They both can contain default methods.
37	C	The code does not compile because the class inherits two default methods with the same signature and no overridden version.
38	A	Protected instance method is a virtual method.
39	B	An interface extends another interface, while a class extends another class.
40	A	The output is "2". The reference type determines the version of the secret variable that is selected.
41	D	Public final void dance() is a valid override by a subclass.
42	C	The answer is new Wolf() because Wolf is not a sub class of Dog.
43	A	Final modifiers cannot be applied to an interface method.
44	A	It compiled and at runtime prints Let's start the party!
45	D	The answer is none of the above.
46	B	If a parent class does not include a no-argument constructor (nor a default one inserted by the compiler), a child class must contain at least one constructor definition.
47	D	The object type determines which attributes exist in memory, while the reference type determines which attributes are accessible by the caller.
48	A	Long is a subclass of Number, and therefore, it is covariant with the version in MusicCreator.
49	B	Add backward compatibility to existing interfaces is the best reason for creating a default interface method.
50	C	The code does not compile. Since IO Exception is a superclass of EOFException, we see that this is a broader exception and therefore not compatible.

Reference

Scott S, Jeanne B.: OCA/OCP Java SE 8 Programmer Practice Tests. Indiana, USA: 2017.