socrative

Answer Key
February 24, 2021

OCA Practice Questions Ch. 6 ('Copy')

1. The ________________ access modifier allows access to everything the ________________ access modifier does and more.

A package-private, protected

B protected, public

✓ C protected, package-private

D private, package-private

i The protected modifier allows access by subclasses and members within the same package, while the package-private modifier allows access only to members in the same package. Therefore, the protected access modifier allows access to everything the package-private access modifier, plus subclasses, making Option C the correct answer. Options A, B, and D are incorrect because the first term is a more restrictive access modifier than the second term.

2. What is the command to call one constructor from another constructor in the same class?

A super()

✓ B this()

C that()

D construct()

i The super() statement is used to call a constructor in a parent class, while the this() statement is used to call a constructor in the same class, making Option B correct and Option A incorrect. Options C and D are incorrect because they are not constructors.

3. What is the output of the following application?

```
package stocks;
public class Bond {
   private static int price = 5;
   public boolean sell() {
      if(price<10) {
         price++;
         return true;
      } else if(price>=10) {
         return false;
      }
   }

   public static void main(String[] cash) {
      new Bond().sell();
      new Bond().sell();
      new Bond().sell();
      System.out.print(price);
   }
}
```

A 5

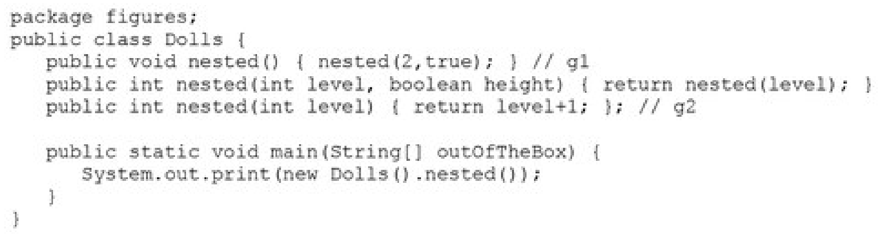
B 6

C 8

✓ D The code does not compile.

i The sell() method does not compile because it does not return a value if both of the if-then statements' conditional expressions evaluate to false. While logically, it is true that price is either less than 10 or greater than or equal to 10, the compiler does not know that. It just knows that if both if-then statements evaluate to false, then it does not have a return value, therefore it does not compile.

4.



```
package figures;
public class Dolls {
   public void nested() { nested(2,true); } // g1
   public int nested(int level, boolean height) { return nested(level); }
   public int nested(int level) { return level+1; }; // g2

   public static void main(String[] outOfTheBox) {
      System.out.print(new Dolls().nested());
   }
}
```

What is true about the following program?

A It compiles successfully and prints 3 at runtime.

B It does not compile because of line g1.

C It does not compile because of line g2.

[✓ D] It does not compile for some other reason.

The three overloaded versions of nested() compile without issue, since each method takes a different set of input arguments, making Options B and C incorrect. The code does not compile, though, due to the first line of the main() method, making Option A incorrect. The no-argument version of the nested() method does not return a value, and trying to output a void return type in the print() method throws an exception at runtime.

5. Java uses _______ to send data into a method.

A pass-by-null

[✓ B] pass-by-value

C both pass-by-value and pass-by-reference

D pass-by-reference

Java uses pass-by-value to copy primitives and references of objects into a method. That means changes to the primitive value or reference in the method are not carried to the calling method. That said, the data within an object can change, just not the original reference itself. Therefore, Option B is the correct answer, and Options C and D are incorrect. Option A is not a real term.

6. Which of the following is a valid JavaBean method signature?

A public void getArrow()

B public void setBow()

[✓ C] public void setRange(int range)

D public String addTarget(String target)

Option A is incorrect because the getter should return a value. Option B is incorrect because the setter should take a value. Option D is incorrect because the setter should start with set and should not return a value. Option C is a correct setter declaration because it takes a value, uses the void return type, and uses the correct naming convention.

7. Which of the following statements about calling this() in a constructor is not true?

A If this() is used, it must be the first line of the constructor.

[✓ B] If super() and this() are both used in the same constructor, super() must appear on the line immediately after this().

C If arguments are provided to this(), then there must be a constructor in the class able to take those arguments.

D If the no-argument this() is called, then the class must explicitly implement the no-argument constructor.

i Options A, C, and D are true statements about calling this() inside a constructor. Option B is incorrect because a constructor can only call this() or super() on the first line of the constructor, but never both in the same constructor. If both constructors were allowed to be called, there would be two separate calls to super(), leading to duplicate initialization of parent constructors, since the other constructor referenced by this() would also call super() (or be chained to one that eventually calls super()).

8.

```
package ai;
public class Robot {
    _______ compute() { return 10; }
}
```

Which of the following can fill in the blank to make the class compile?

A Public int

[✓ B] Long

C void

D private String

i Option A is incorrect because the public access modifier starts with a lowercase letter. Options C and D are incorrect because the return types, void and String, are incompatible with the method body that returns an integer value of 10. Option B is correct and has package-private access. It also uses a return type of Long that the integer value of 10 can be easily assigned to without an explicit cast.

9. A _______________ variable is always available to all instances of the class.

A public

B local

[✓ C] static

D instance

i The only variables always available to all instances of the class are those declared static; therefore, Option C is the correct answer. Option A may seem correct, but public variables are only available if a reference to the object is maintained among all instances. Option B is incorrect because there is no local keyword in Java. Option D is also incorrect because a private instance variable is only accessible within the instance that created it.

10. Which line of code, inserted at line p1, causes the application to print 5?

```
package games;
public class Jump {
   private int rope = 1;
   protected boolean outside;

   public Jump() {
      // p1
      outside = true;
   }

   public Jump(int rope) {
      this.rope = outside ? rope : rope+1;
   }

   public static void main(String[] bounce) {
      System.out.print(new Jump().rope);
   }
}
```

[✓ **A**] this(4);

B new Jump(4);

C this(5);

D rope = 4;

First off, all of the lines compile but they produce various different results. Remember that the default initialization of a boolean instance variable is false, making outside false at line p1. Therefore, this(4) will cause rope to be set to 5, while this(5) will cause rope to be set to 6. Since 5 is the number we are looking for, Option A is correct, and Option C is incorrect. Option B is incorrect. While the statement does create a new instance of Jump, with rope having a value of 5, that instance is nested and the value of rope does not affect the surrounding instance of Jump that the constructor was called in. Option D is also incorrect. The value assigned to rope is 4, not the target 5.

11. Which of the following statements is not true?

A An instance of one class may access an instance of another class's attributes if it has a reference to the instance and the attributes are declared public.

[✓ **B**] An instance of one class may access package-private attributes in a parent class, provided the parent class is not in the same package.

C Two instances of the same class may access each other's private attributes.

D An instance of one class may access an instance of another class's attributes if both classes are located in the same package and marked protected.

Options A, C, and D are true statements. In particular, Option C allows us to write the equals() methods between two objects that compare private attributes of the class. Option D is true because protected access also provides package-private access. Option B is false. Package-private attributes are only visible if the two classes are in the same package, regardless of whether one extends the other.

12. Given the following class, what should be inserted into the two blanks to ensure the class data is properly encapsulated?

```
package storage;
public class Box {
   public String stuff;

   ______ String ______() {
      return stuff;
   }

   public void setStuff(String stuff) {
      this.stuff = stuff;
   }
}
```

A public and getStuff

B private and isStuff

C public and setStuff

[✓ **D**] None of the above

The class data, stuff, is declared public, allowing any class to modify the stuff variable and making the implementation inherently unsafe for encapsulation. Therefore, there are no values that can be placed in the two blanks to ensure the class properly encapsulates its data, making Option D correct. Note that if stuff was declared private, Options A, B, and C would all be correct. Encapsulation does not require JavaBean syntax, just that the internal attributes are protected from outside access, which all of these sets of values do achieve.

13. Which statement about a no-argument constructor is true?

A The Java compiler will always insert a default no-argument constructor if you do not define a no-argument constructor in your class.

B In order for a class to call super() in one of its constructors, its parent class must explicitly implement a no-argument constructor.

✓ **C** If a class extends another class that has only one constructor that takes a value, then the child class must explicitly declare at least one constructor.

D A class may contain more than one no-argument constructor.

i Option A is incorrect because Java only inserts a no-argument constructor if there are no other constructors in the class. Option B is incorrect because the parent can have a default no-argument constructor, which is inserted by the compiler and accessible in the child class. Finally, Option D is incorrect. A class that contains two no-argument constructors will not compile because they would have the same signature. Finally, Option C is correct. If a class extends a parent class that does not include a no-argument constructor, the default no-argument constructor cannot be automatically inserted into the child class by the compiler. Instead, the developer must explicitly declare at least one constructor and explicitly define how the call to the parent constructor is made.

14. Which of the following method signatures does not contain a compiler error?

✓ **A** public void sing(String key, String... harmonies)

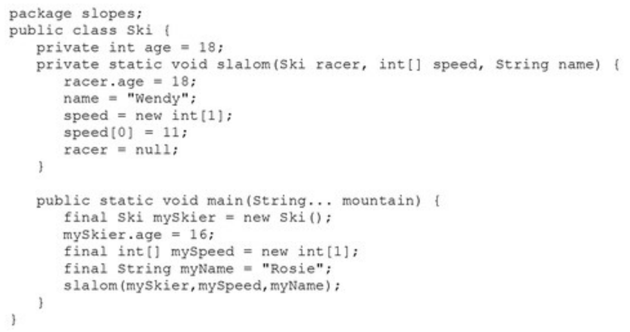
B public void sing(int note, String... sound, int music)

C public void sing(String... keys, String... pitches)

D public void sing(String... notes, String melodies)

i A method may contain at most one varargs parameter, and it must appear as the last argument in the list. For this reason, Option A is correct, and Options B, C, and D are incorrect.

15.



```java
package slopes;
public class Ski {
   private int age = 18;
   private static void slalom(Ski racer, int[] speed, String name) {
      racer.age = 18;
      name = "Wendy";
      speed = new int[1];
      speed[0] = 11;
      racer = null;
   }

   public static void main(String... mountain) {
      final Ski mySkier = new Ski();
      mySkier.age = 16;
      final int[] mySpeed = new int[1];
      final String myName = "Rosie";
      slalom(mySkier,mySpeed,myName);
   }
}
```

Given the following application, what is the value of mySkier.age in the main() method after the call to the slalom() method?

A null

B 16

✓ **C** 18

D 0

i To solve this problem, it helps to remember that Java is a pass-by-value language in which copies of primitives and object references are sent to methods. This also means that an object's data can be modified within a method and shared with the caller, but not the reference to the object. Any changes to the object's reference within the method are not carried over to the caller. In the slalom() method, the Ski object is updated with an age value of 18. Although, the last line of the slalom() method changes the variable value to null, it does not affect the mySkier object or reference in the main() method. Therefore, the mySkier object is not null and the age variable is set to 18, making Options A and D incorrect and option C correct.

16.

```
public class Calculations {
   public Integer findAverage(int sum) { return sum; }
}
```

Given the class below, which method signature could be successfully added to the class as an overloaded version of the findAverage() method?

A public Long findAverage(int sum)

[✓ B] public Long findAverage(int sum, int divisor)

C public Integer average(int sum)

D private void findAverage(int sum)

i Options A and D would not allow the class to compile because two methods in the class cannot have the same name and arguments, but a different return value. Option C would allow the class to compile, but it is not a valid overloaded form of our findAverage() method since it uses a different method name. Option B is a valid overloaded version of the findAverage() method, since the name is the same but the argument list differs.

17. Which of the following is not a reason to use encapsulation when designing a class?

A Promote usability by other developers

B Maintain class data integrity of data elements

C Prevent users from modifying the internal attributes of a class

[✓ D] Increase concurrency and improve performance

i Implementing encapsulation prevents internal attributes of a class from being modified directly, so Option C is a true statement. By preventing access to internal attributes, we can also maintain class data integrity between elements, making Option B a true statement. Option A is also a true statement about encapsulation, since well-encapsulated classes are often easier to use. Option D is an incorrect statement. Encapsulation makes no guarantees about performance and concurrency.

18. Which of the following data types can be modified after they are passed to a method as an argument?

[✓ A] int[]

B String

C long

D boolean

i Option B is incorrect because String values are immutable and cannot be modified. Options C and D are also incorrect since variables are passed by value, not reference, in Java. Option A is the correct answer.

The contents of an array can be modified when passed to a method, since a copy of the reference to the object is passed. For example, the method can change the first element of a non-empty array.

19.

```
package useful;
public class MathHelper {
   public static int roundValue(double d) {
      // Implementation omitted
   }
}
```

What is the best way to call the following method from another class in the same package, assuming the class using the method does not have any static imports?

- **A** MathHelper:roundValue(5.92)
- ✓ **B** MathHelper.roundValue(3.1)
- **C** roundValue(4.1)
- **D** useful.MathHelper.roundValue(65.3)

i Option A is not a valid syntax in Java. Option C would be correct if there was a static import, but the question specifically says there are not any. Option D is almost correct, since it is a way to call the method, but the question asks for the best way to call the method. In that regard, Option B is the best way to call the method, since we are given that two classes are in the same package, therefore the package name would not be required.

20. Given a method with one of the following return types, which data type prevents the return statement from being used within the method?

- **A** byte
- **B** String
- **C** void
- ✓ **D** None of the above

i Options A and B are incorrect because a method with a non-void return type requires that the method return a value using the return statement. Option C is also incorrect since a method with a void return type can still call the return command with no values and exit the method. Therefore, Option D is the correct answer.

21.

```
package end;
public final class Games {
   public final static int finish(final int score) {
      final int win = 3;
      final int result = score++ < 5 ? 2 : win;
      return result+=win;
   }
   public static void main(final String[] v) {
      System.out.print(finish(Integer.parseInt(v[0])));
   }
}
```

How many final modifiers would need to be removed for this application to compile?

A None

B One

[✓ C] Two

D The code will not compile regardless of the number of final modifiers that are removed.

i The finish() method modifies two variables that are marked final, score and result. The score variable is modified by the post-increment ++ operator, while the result variable is modified by the compound addition += operator. Removing both final modifiers allows the code to compile. For this reason, Option C is the correct answer.

22. _________________ is used to call a constructor in the parent class, while _________________ is used to reference a member of the parent class.

A super and this()

B super and super()

C super() and this

[✓ D] super() and super

i The super() statement is used to call a constructor in the parent class, while super is used to reference a member of the parent class. The this() statement is used to call a constructor in the current class, while this is used to reference a member of the current class. For these reasons, Option D is the correct answer.

23.

```
void run(String government)
```

Given the following method signature, which classes can call it?

A Classes in other packages

[✓ B] Classes in the same package

C Subclasses in a different package

D All classes

i The method signature has package-private, or default, access; therefore, it is accessible to classes in the same package, making Option B the correct answer.

24.

```
package shield;
public class Protect {
   private String material;
   protected int strength;

   public int getStrength() {
      return strength;
   }
   public void setStrength(int strength) {
      this.strength = strength;
   }
}
```

Which statement(s) about the following class would help to properly encapsulate the data in the class?

I. Change the access modifier of strength to private.
II. Add a getter method for material.
III. Add a setter method for material.

[✓ A] I

B II and III

C I, II, and III

D None, the data in the class is already encapsulated.

i The access modifier of strength is protected, meaning subclasses and classes within the same package can modify it. Changing the value to private would improve encapsulation by making the Protect class the only one capable of directly modifying it. For these reasons, the first statement is correct. Alternatively, the second and third statements do not improve the encapsulation of the class. While having getters and setters for private variables is helpful, they are not required. Encapsulation is about protecting the data elements. With this in mind, it is clear the material variable is already protected. Therefore, Option A is the correct answer.

25. Which of the following is a valid method name in Java?

[✓ A] Go_$Outside$2()

B have-Fun()

C new()

D 9enjoyTheWeather()

i Option A is correct since method names may include the underscore _ character as well as the dollar $ symbol. Note that there is no rule that requires a method start with a lowercase character; it is just a practice adopted by the community. Option B is incorrect because the hyphen - character may not be part of a method name. Option C is incorrect since new is a reserved word in Java. Finally, Option D is incorrect. A method name must start with a letter, the dollar $ symbol, or an underscore _ character.

26.

```
package farm;
public class Coop {
   public final int static getNumberOfChickens() {
      // INSERT CODE HERE
   }
}
```

Which of the following lines of code can be inserted in the line below that would allow the class to compile?

- **A** return 3.0;
- **B** return 5L;
- **C** return 10;
- ✓ **D** None of the above

i The code does not compile, regardless of what is inserted into the line because the method signature is invalid. The return type, int, should go before the method name and after any access, final, or static modifiers. Therefore, Option D is the correct answer. If the method was fixed, by swapping the order of int and static in the method declaration, then Option C would be the correct answer. Options A and B are still incorrect, though, since each uses a return type that cannot be implicitly converted to int.

27. Which of the following is a true statement about passing data to a method?

- **A** A change made to a primitive value passed to a method is reflected in the calling method.
- ✓ **B** A change made to the data within an object passed to a method is reflected in the calling method.
- **C** Reassigning an object reference passed to a method is reflected in the calling method.
- **D** A change made to a boolean value passed to a method is reflected in the calling method.

i Java uses pass-by-value, so changes made to primitive values and object references passed to a method are not reflected in the calling method. For this reason, Options A and C are incorrect statements. Option D is also an invalid statement because it is a special case of Option A. Finally, Option B is the correct answer. Changes to the data within an object are visible to the calling method since the object that the copied reference points to is the same.

28. What is a possible output of the following application?

- **A** Your gift: wrap.Gift@29ca2745
- **B** Your gift: Your gift:
- ✓ **C** It does not compile.
- **D** It compiles but throws an exception at runtime.

```
package wrap;
public class Gift {
   private final Object contents;
   protected Object getContents() {
      return contents;
   }
   protected void setContents(Object contents) {
      this.contents = contents;
   }
   public void showPresent() {
      System.out.print("Your gift: "+contents);
   }
   public static void main(String[] treats) {
      Gift gift = new Gift();
      gift.setContents(gift);
      gift.showPresent();
   }
}
```

i The code contains a compilation problem in regard to the contents instance variable. The contents instance variable is marked final, but there is a setContents() instance method that can change the value of the variable. Since these two are incompatible, the code does not compile, and Option C is correct. If the final modifier was removed from the contents variable declaration, then the expected output would be of the form shown in Option A.

29. Which of the following is a valid JavaBean method prefix?

A is

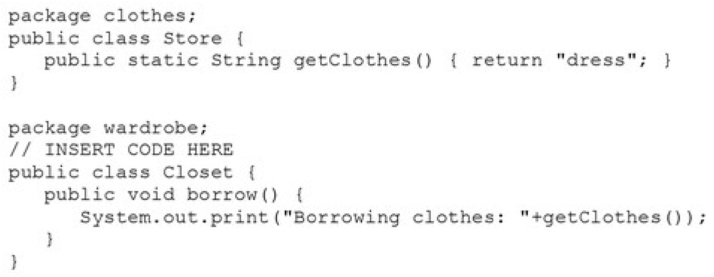
B gimme

C request

D put

i JavaBean methods use the prefixes get, set, and is for boolean values, making Option A the correct choice.

30.



```
package clothes;
public class Store {
   public static String getClothes() { return "dress"; }
}

package wardrobe;
// INSERT CODE HERE
public class Closet {
   public void borrow() {
      System.out.print("Borrowing clothes: "+getClothes());
   }
}
```

Given the following two classes, each in a different package, which line inserted below allows the second class to compile?

A static import clothes.Store.getClothes;

B import clothes.Store.*;

[✓ C] import static clothes.Store.getClothes;

D

import static clothes.Store;

i Option A is incorrect because the keywords static and import are reversed. The Closet class uses the method getClothes() without a reference to the class name Store, therefore a static import is required. For this reason, Option B is incorrect since it is missing the static keyword. Option D is also incorrect since static imports are used with members of the class, not a class name. Finally, Option C is the correct answer since it properly imports the method into the class using a static import.

31. What access modifier is used to mark class members package-private?

A private

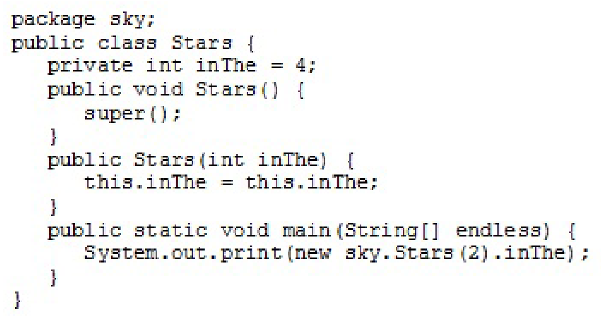
B default

C protected

[✓ D] None of the above

i In Java, the lack of an access modifier indicates that the member is package-private, therefore Option D is correct. Note that the default keyword is used for interfaces and switch statements, and is not an access modifier.

32.



```java
package sky;
public class Stars {
   private int inThe = 4;
   public void Stars() {
      super();
   }
   public Stars(int inThe) {
      this.inThe = this.inThe;
   }
   public static void main(String[] endless) {
      System.out.print(new sky.Stars(2).inThe);
   }
}
```

How many lines of the following program contain compilation errors?

A None

[✓ B] One

C Two

D Three

i The code does not compile, so Option A is incorrect. The class contains two constructors and one method. The first method, Stars(), looks a lot like a no-argument constructor, but since it has a return value of void, it is a method, not a constructor. Since only constructors can call super(), the code does not compile due to this line. The only constructor in this class, which takes an int value as input, performs a pointless assignment, assigning a variable to itself. While this assignment has no effect, it does not prevent the code from compiling. Finally, the main() method compiles without issue since we just inserted the full package name into the class constructor call. This is how a class that does not use an import statement could call the constructor. Since the method is in the same class, and therefore the same package, it is redundant to include the package name but not disallowed. Because only one line causes the class to fail to compile, Option B is correct.

33. Which of the following statements is true?

[✓ A] An instance method is allowed to reference a static variable.

B A static method is allowed to reference an instance variable.

C A static initialization block is allowed to reference an instance variable.

D A final static variable may be set in a constructor.

i An instance method or constructor has access to all static variables, making Option A correct. On the other hand, static methods and static initializers cannot reference instance variables since they are defined across all instances, making Options B and C incorrect. Note that they can access instance variables if they are passed a reference to a specific instance, but not in the general case. Finally, Option D is incorrect because static final variables must be set when they are declared or in a static initialization block.

34.

```
public short calculateDistance(double lat1, double lon1,
      double lat2, double lon2) {
   // INSERT CODE HERE
}
```

Given the following method declaration, which line can be inserted to make the code compile?

- **A** return new Integer(3);
- [✓ **B**] return new Byte((byte)6);
- **C** return 5L;
- **D** return new Short(4).longValue();

i The method calculateDistance() requires a return type that can be easily converted to a short value. Options A, C, and D are incorrect because they each use a larger data type that requires an explicit cast. Option D also does not compile because the Short constructor requires an explicit cast to convert the value of 4, which is assumed to be an int, to a short, as shown in new Short((short)4). Option B is the correct answer since a byte value can be easily promoted to short and returned by the method.

35. Which of the following statements about overloaded methods are true?
I. Overloaded methods must have the same name.
II. Overloaded methods must have the same return type.
III. Overloaded methods must have a different list of parameters.

- **A** I
- **B** I and II
- [✓ **C**] I and III
- **D** I, II, and III

i Overloaded methods have the same name but a different list of parameters, making the first and third statements true. The second statement is false, since overloaded methods can have the same or different return types. Therefore, Option C is the correct answer.

36.

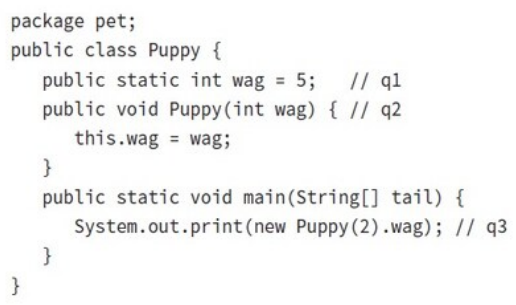
```
package work;
public class Week {
   private static final String monday;
   String tuesday;
   final static wednesday = 3;
   final protected int thursday = 4;
}
```

How many lines of code would need to be removed for the following class to compile?

- **A** One
- [✓ **B**] Two
- **C** Three
- **D** The code will not compile regardless of the number of lines removed.

i The declaration of monday does not compile, because the value of a static final variable must be set when it is declared or in a static initialization block. The declaration of tuesday is fine and compiles without issue. The declaration of wednesday does not compile because there is no data type for the variable. Finally, the declaration of thursday does compile because the final modifier can appear before the access modifier. For these reasons, Option B is the correct answer.

37.



```
package pet;
public class Puppy {
   public static int wag = 5;   // q1
   public void Puppy(int wag) { // q2
      this.wag = wag;
   }
   public static void main(String[] tail) {
      System.out.print(new Puppy(2).wag); // q3
   }
}
```

What is the output of the following application?

A 2

B It does not compile because of line q1.

C It does not compile because of line q2.

✓ D It does not compile because of line q3.

i The Puppy class does not declare a constructor, so the default no-argument constructor is automatically inserted by the compiler. What looks like a constructor in the class is actually a method that has a return type of void. Therefore, the line in the main() method to create the new Puppy(2) object does not compile, since there is no constructor capable of taking an int value, making Option D the correct answer.

38. The ________________ access modifier allows access to everything the ________________ access modifier does and more.

✓ A public, private

B private, package-private

C package-private, protected

D private, public

i The public modifier allows access to members in the same class, package, subclass, or even classes in other packages, while the private modifier allows access only to members in the same class. Therefore, the public access modifier allows access to everything the private access modifier does, and more, making Option A the correct answer. Options B, C, and D are incorrect because the first term is a more restrictive access modifier than the second term.

39.

```
package ship;
public class Phone {
   private int size;
   public Phone(int size) {this.size=size;}

   public static void sendHome(Phone p, int newSize) {
      p = new Phone(newSize);
      p.size = 4;
   }
   public static final void main(String... params) {
      final Phone phone = new Phone(3);
      sendHome(phone,7);
      System.out.print(phone.size);
   }
}
```

What is the output of the following application?

[✓ A] 3

B 4

C 7

D The code does not compile.

i The code compiles without issue, so Option D is incorrect. The key here is that java uses pass-by-value to send object references to methods. Since the Phone reference p was reassigned in the first line of the sendHome() method, any changes to the p reference were made to a new object. In other words, no changes in the sendHome() method affected the object that was passed in. Therefore, the value of size was the same before and after the method call, making the output 3 and Option A the correct answer.

40.

```
public class Drink {
   public static void water() {}
   public void get() {
      // INSERT CODE HERE
   }
}
```

Given the following class, which line of code when inserted below would prevent the class from compiling?

A water();

[✓ B] this.Drink.water();

C this.water();

D Drink.water();

i Options A and D are equivalent and would allow the code to compile. They both are proper ways to access a static method from within an instance method. Option B is the correct answer. The class would not compile because this.Drink has no meaning to the compiler. Finally, Option C would still allow the code to compile, even though it is considered a poor coding practice. While static members should be accessed in a static way, it is not required.

41.

```
public void call(int count, String me, String... data)
```

Given the following method declaration signature, which of the following is a valid call of this method?

A call(9,"me",10,"AI")

B call(5)

[✓ C] call(2,"home","sweet")

D call("answering","service")

> i The method signature requires one int value, followed by exactly one String, followed by String varargs, which can be an array of String values or zero or more individual String values. Only Option C conforms to these requirements, making it the correct answer.

42. Which statement about a static variable is true?

A The value of a static variable must be set when the variable is declared or in a static initialization block.

B It is not possible to read static final variables outside the class in which they are defined.

C It is not possible to reference static methods using static imports.

[✓ D] A static variable is always available in all instances of the class.

> i Option A is a statement about final static variables, not all static variables. Option B only applies to static variables marked private, not final. Option C is false because static imports can be used to reference both variables and methods. Option D is the correct answer because a static variable is accessible to all instances of the class.

43. Which of the following is not a true statement?

[✓ A] The first line of every constructor is a call to the parent constructor via the super() command.

B A class does not have to have a constructor explicitly defined.

C A constructor may pass arguments to the parent constructor.

D A final instance variable whose value is not set when they are declared or in an initialization block should be set by the constructor.

> i Option A is the correct answer because the first line of a constructor could be this() or super(), making it an untrue statement. Option B is a true statement because the compiler will insert the default no-argument constructor if one is not defined. Option C is also a true statement, since zero or more arguments may be passed to the parent constructor, if the parent class defines such constructors. Option D is also true. The value of a final instance variable should be set when it is declared, in an initialization block, or in a constructor.

44. How many final modifiers would need to be removed for this application to compile?

```
package park;
public class Tree {
   public final static long numberOfTrees;
   public final double height;
   static {}
   { final int initHeight = 2;
     height = initHeight;
   }
   static {
      numberOfTrees = 100;
      height = 4;
   }
}
```

A None

B One

C Two

✓ D The code will not compile regardless of the number of final modifiers removed.

i The last static initialization block accesses height, which is an instance variable, not a static variable. Therefore, the code will not compile no matter how many final modifiers are removed, making Option D the correct answer. Note that if the line height = 4; was removed, then no final modifiers would need to be removed to make the class compile.

45.

```
package jungle;
public class RainForest extends Forest {
   public RainForest(long treeCount) {
      this.treeCount = treeCount+1;
   }
   public static void main(String[] birds) {
      System.out.print(new RainForest(5).treeCount);
   }
}
class Forest {
   public long treeCount;
   public Forest(long treeCount) {
      this.treeCount = treeCount+2;
   }
}
```

What is the output of the following application?

A 5

B 6

C 8

✓ D The code does not compile.

i Since a constructor call is not the first line of the RainForest() constructor, the compiler inserts the no-argument super() call. Since the parent class, Forest, does not define a no-argument super() constructor, the RainForest() constructor does not compile, and Option D is correct.

46.

```
public class ChooseWisely {
   public ChooseWisely() { super(); }
   public int choose(int choice) { return 5; }
   public int choose(short choice) { return 2; }
   public int choose(long choice) { return 11; }
   public static void main(String[] path) {
      System.out.print(new ChooseWisely().choose((byte)2+1));
   }
}
```

What is the output of the following application?

✓ A 5

B 2

C 11

D The code does not compile.

i

The code compiles without issue, so Option D is incorrect. In the main() method, the value 2 is first cast to a byte. It is then increased by one using the addition + operator. The addition + operator automatically promotes all byte and short values to int. Therefore, the value passed to the choose() in the main() method is an int. The choose(int) method is called, returning 5 and making Option A the correct answer. Note that without the addition operation in the main() method, byte would have been used as the parameter to the choose() method, causing the choose(short) to be selected as the next closest type and outputting 2, making Option B the correct answer.

47.

```
package sports;
public class Football {
   public static Long getScore(Long timeRemaining) {
      return 2*timeRemaining; // m1
   }

   public static void main(String[] refs) {
      final int startTime = 4;
      System.out.print(getScore(startTime)); // m2
   }
}
```

What is the output of the following application?

A 8

B The code does not compile because of line m1.

✓ C The code does not compile because of line m2.

D The code compiles but throws an exception at runtime.

i The variable startTime can be automatically converted to Integer by the compiler, but Integer is not a subclass of Long. Therefore, the code does not compile due the wrong variable type being passed to the getScore() method on line m2, and Option C is correct.

48. Which of the following is a valid method name in Java?

✓ A $sprint()

B \jog13()

C walk#()

D %run()

i Java methods must start with a letter, the dollar $ symbol, or underscore _ character. For these reasons, Options B and D are incorrect, and Option A is correct. Option C is incorrect. The hashtag (#) symbol cannot be included in a method name.

49. Assume there is a class Bouncer with a protected variable. Methods in which class can access this variable?

A Only subclasses of Bouncer

✓ B Any subclass of Bouncer or any class in the same package as Bouncer

C Only classes in the same package as Bouncer

D Any superclass of Bouncer

i The protected modifier allows access by any subclass or class that is in the same package, therefore Option B is the correct answer.

50.

```
package commerce;
public class Bank {
   public void withdrawal(int amountInCents) {}
   public void deposit(int amountInCents) {}
}

package employee;
// INSERT CODE HERE
public class Teller {
   public void processAccount(int depositSlip, int withdrawalSlip) {
      withdrawal(withdrawalSlip);
      deposit(depositSlip);
   }
}
```

Given the following two classes, each in a different package, which line inserted below allows the second class to compile?

A import static commerce.Bank.*;

B static import commerce.Bank.*;

C import static commerce.Bank;

✓ **D** None of the above

i A static import is used to import static members of another class. In this case, the withdrawal() and deposit() methods in the Bank class are not marked static. They require an instance of Bank to be used and cannot be imported as static methods. Therefore, Option D is correct. If the two methods in the Bank class were marked static, then Option A would be the correct answer since wildcards can be used with static imports to import more than one method. Option B reverses the keywords static and import, while Option C incorrectly imports a class, which cannot be imported via a static import.