1. What is the output of following program:-

```
public class Airplane
{
static int start = 2;
final int end;
public Airplane(int x)
{
x = 4;
end = x;
}
public void fly(int distance) {
System.out.print(end-start+" ");
System.out.print(distance);
}
public static void main(String... start) {
new Airplane(10).fly(5);
}
}
```

A.2 5

B.8 5

C.6 5

D.The code does not compile.

A The code compiles so Option D is incorrect. The input to the constructor is ignored, making the assignment of end to be 4. Since start is 2, the subtraction of 4 by 2 results in the application printing 2, followed by 5, making Option A the correct answer.

2.What is the output of the following code snippet?

```
String tree = "pine";
int count = 0;
if (tree.equals("pine")) {
int height = 55;
count = count + 1;
}
System.out.print(height + count);
```

A. 1

B. 55

C. 56

D. It does not compile.

D. The height variable is declared within the if-then statement block. Therefore, it cannot be referenced outside the if-then statement and the code does not compile.

3.Given that a Math class exists in both the java.lang and pocket.complex packages, what is the result of compiling the following class?

```
1: package pocket;
2: import pocket.complex.*;
3: import java.util.*;
4: public class Calculator {
5:    public static void main(String[] args) {
6:          System.out.print(Math.floor(5));
7:    }
8: }
```

A. The code does not compile because of line 2.

B. The code does not compile because of line 3.

C. The code does not compile because of line 6.

D. The code compiles without issue.

C. Remember that java.lang is automatically imported in all Java classes, therefore both java.lang.Math and pocket.complex.Math are both imported into this class. Importing both sets of packages does not cause any compilation issues, making Option A incorrect. Line 3 is unnecessary import but including it does not prevent the class from compiling, making Option B incorrect. While both versions of Math may be imported into the class, the usage of the Math class requires a package name. Because of this, line 6 does not compile as the class reference is ambiguous, making Option C the correct answer and Option D incorrect.

4.What is the output of the following?

```
public static void main(String... args) {
String chair, table = "metal";
chair = chair + table;
System.out.println(chair);
}
```

A.metal

B.metalmetal

C.nullmetal

D.The code does not compile.

D. The table variable is initialized to "metal". However, chair is not initialized. In Java, initialization is per variable and not for all the variables in a single declaration. Therefore, the second line tries to reference an uninitialized local variable and does not compile, which makes Option D correct.

5.How many of the following methods compile?

```
public String convert(int value) {
return value.toString();
}
public String convert(Integer value) {
```

```
return value.toString();
}
public String convert(Object value) {
return value.toString();
}
```

A. None

B. One

C. Two

D. Three

C. Objects have instance methods while primitives do not. Since int is a primitive, you cannot call instance methods on it. Integer and String are both objects and have instance methods. Therefore, Option C is correct.

6.What is the result of running this code?

```
public class Values {
integer a = Integer.valueOf("1");
public static void main(String[] nums) {
integer a = Integer.valueOf("2");
integer b = Integer.valueOf("3");
System.out.println(a + b);
}
}
```

A. 4

B. 5

C. The code does not compile.

D. The code compiles but throws an exception at runtime.

C. There is no class named integer. There is a primitive int and a class Integer. There- fore, the code does not compile, and Option C is correct. If the type was changed to Integer, Option B would be correct.

7.Which of the following is a wrapper class?

A. int

B. Int

C. Integer

D. Object

C. Option A is incorrect because int is a primitive. Option B is incorrect because it is not the name of a class in Java. While Option D is a class in Java, it is not a wrapper class because it does not map to a primitive. Therefore, Option C is correct.

8.How many instance initializers are in this code?

```
1: public class Bowling {
2:    { System.out.println(); }
3:    public Bowling () {
4:    System.out.println();
5:    }
6:    static { System.out.println(); }
7:    { System.out.println(); }
8: }
```

A. None

B. One

C. Two

D. Three

C. Lines 2 and 7 illustrate instance initializers. Line 6 is a static initializer. Lines 3–5 are a constructor.

9.What is the first line in the following code to not compile?

```
public static void main(String[] args) {
int Integer = 0; // k1
Integer int = 0; // k2
Integer ++;      // k3
int++;     // k4
}
```

A. k1

B. k2

C. k3

D. k4

B. Integer is the name of a class in Java. While it is bad practice to use the name of a class as your local variable name, this is legal. Therefore, k1 does compile. It is not legal to use a reserved word as a variable name. All of the primitives including int are reserved words. Therefore, k2 does not compile, and Option B is the answer. Line k4 doesn't compile either, but the question asks about the first line to not compile.

10.Which of the following can fill in the blanks to make this code compile?

```
_________d = new _________(1_000_000_.00);
```

A. double, double

B. double, Double

C. Double, double

D. None of the above

D. This question is tricky as it appears to be about primitive vs. wrapper classes. Looking closely, there is an underscore right before the decimal point. This is illegal as the under- score in a numeric literal can only appear between two digits.

11.Given the following code, fill in the blank to have the code print bounce.

```
public class TennisBall {
public TennisBall() {
System.out.println("bounce");
}
public static void main(String[] slam) {
_____________________________________
}
}
```

A.TennisBall;

B.TennisBall();

C.new TennisBall;

D.new TennisBall();

D. In order to call a constructor, you must use the new keyword. It cannot be called as if it was a normal method. This rules out Options A and B. Further, Option C is incorrect because the parentheses are required.

12.Which of the following can fill in the blanks to make this code compile?

```
--------------d = new ------------(1_000_000.00);
```

A. double, double

B. double, Double

C. Double, double

D. None of the above

B. A constructor can only be called with a class name rather than a primitive, making Options A and C incorrect. The newly constructed Double object can be assigned to either a double or Double thanks to autoboxing. Therefore, Option B is correct.

13.What does the following output?

```
1: public class InitOrder {
2:    public String first = "instance";
3:    public InitOrder() {
4:    first = "constructor";
5:    }
6:    { first = "block"; }
7:    public void print() {
8:    System.out.println(first);
9:    }
10:   public static void main(String... args) {
11:   new InitOrder().print();
12:   }
13: }
```

A. block

B. constructor

C. instance

D. The code does not compile.

B. First line 2 runs and sets the variable using the declaration. Then the instance initializer on line 6 runs. Finally, the constructor runs. Since the constructor is the last to run of the three, that is the value that is set when we print the result, so Option B is correct.

14.Which is the most common way to fill in the blank to implement this method?

```
public class Penguin {
private double beakLength;
```

```
public static void setBeakLength(Penguin p, int b)
{
-------------------------------------
}
}
```

A. p.beakLength = b;

B. p['beakLength'] = b;

C. p[beakLength] = b;

D. None of the above

A. Options B and C do not compile. In Java, braces are for arrays rather than instance vari- ables. Option A is the correct answer. It uses dot notation to access the instance variable. It also shows that a private variable is accessible in the same class and that a narrower type is allowed to be assigned to a wider type.

15.Fill in the blanks to indicate whether a primitive or wrapper class can be assigned without the compiler using the autoboxing feature.

```
----------first = Integer.parseInt("5");
----------second = Integer.valueOf("5");
```

A. int, int

B. int, Integer

C. Integer, int

D. Integer, Integer

B. The parseInt() methods return a primitive. The valueOf() methods return a wrap- per class object. In real code, autoboxing would let you assign the return value to either a primitive or wrapper class. In terms of what gets returned directly, Option B is correct.

16.What is the output of the following?

```
package beach;
public class Sand {
public Sand() {
System.out.print("a");
}
public void Sand() {
System.out.print("b");
}
public void run() {
new Sand(); Sand();
}
public static void main(String... args) {
new Sand().run();
}
}
```

A.a

B.ab

C.aab

D.None of the above

C. The main() method calls the constructor which outputs a. Then the main method calls the run() method. The run() method calls the constructor again, which outputs a again. Then the run() method calls the Sand() method, which happens to have the same name as the constructor. This outputs b. Therefore, Option C is correct.

17.What is the output of the following code snippet?

```
int x = 10, y = 5;
boolean w = true, z = false;
x = w ? y++ : y--;
w = !z;
System.out.print((x+y)+" "+(w ? 5 : 10));
```

A.The code does not compile.

B.10 10

C.11 5

D.12 5

C. The code compiles, so Option A is incorrect. Since w starts out true, the third line takes the first right-hand side of the ternary expression returning and assigning 5 to x (post- increment operator) while incrementing y to 6. Note that the second right-hand side of the ternary expression y-- is not evaluated since ternary operators only evaluate one right-hand expression at runtime. On the fourth line, the value of w is set to !z. Since z is false, the value of w remains true. The final line outputs the value of (5+6) and (true ? 5 : 10), which is 11 5, making Option C the correct answer.

18.What is the value of 12 + 6 * 3 % (1 + 1) in Java?

A.0

B.12

C.14

D.None of the above

B. The question is about operator precedence and order of operation. The multiplication * and modulus % operators have the highest precedence, although what is inside the parentheses needs to be evaluated first. We can reduce the expression to the following: 12 + 6 * 3 % 2. Since

multiplication * and modulus % have the same operator precedence, we evaluate them from left to right as follows: 12 + 6 * 3 % 2 → 12 + 18 % 2 → 12 + 0 → 12. We see that despite all of the operators on the right-hand side of the expression, the result is zero, leaving us a value of 12, making Option B the correct answer.

19.Which of the following is not a possible result of executing the following application?

```
public class ConditionallyLogical {

public static void main(String... data) {

if(data.length>=1 && (data[0].equals("sound") || data[0].equals
("logic")) && data.length<2)

{

System.out.print(data[0]);

}

}

}
```

A.Nothing is printed.

B.sound is printed.

C.The application throws an exception at runtime.

D.logic is printed.

C. The key to understanding this question is to remember that the conditional conjunction && operator only executes the right-hand side of the expression if the left-hand side of the expression is true. If data is an empty array, then the expression ends early and nothing is output. The second part of the expression will return true if data's first element is sound or logic. Since we know from the first part of the statement that data is of length at least one, no exception will be thrown. The final part of the expression with data.length<2 doesn't change the output when data is an array of size one. Therefore, sound

and logic are both possible outputs. For these reasons, Option C is the only result that is unexpected at runtime.

20.What is the output of the following application?

```
public class CountEntries {
public static int getResult(int threshold) {
return threshold > 5 ? 1 : 0;
}

public static final void main(String[] days)
{
System.out.print(getResult(5)+getResult(1)+getResult(0)+getResult(2)+"
");
}
}
```

A.0

B.1

C.0000

D.1000

A. All of the terms of getResult() in this question evaluate to 0, since they are all less than or equal to 5. The expression can therefore be reduced to 0+0+0+0+"". Since Java evaluates the + operator from left to right, the four operands on the left are applied using numeric addition, resulting in the expression 0+"". This expression just converts the value to a String, resulting in an output of 0, making Option A the correct answer.

21.What is the output of the following application?

```
package yoyo;
public class TestGame {
public String runTest(boolean spinner, boolean roller) {
if(spinner = roller) return "up";
else return roller ? "down" : "middle";
}
public static final void main(String pieces[]) {
final TestGame tester = new TestGame();
System.out.println(tester.runTest(false,true));
}
}
```

A. up

B. middle

C. down

D. The code does not compile.

A. The code compiles without issue, so Option D is incorrect. The key here is that the if- then statement in the runTest() method uses the assignment operator (=) instead of the (==) operator. The result is that spinner is assigned a value of true, and the statement (spinner = roller) returns the newly assigned value. The method then returns up, mak- ing Option A the correct answer. If the (==) operator had been used in the if-then statement, then the process would have branched to the else statement, with down being returned by the method.

22.Fill in the blanks: 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

C. The protected modifier allows access by subclasses and members within the same pack- age, 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.

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

A.super()

B.this()

C.that()

D.construct()

B. 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.

24.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.

D. 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.

25.What is true about the following program?

```
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());
```

```
}
}
```

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.

D. 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.

26.Fill in the blank: 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

B. 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 call- ing method. That said, the data within an object can change, just not the original refer- ence itself. Therefore, Option B is the correct answer, and Options C and D are incorrect. Option A is not a real term.

27.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)

C. 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.

28.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.

B. 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()).

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

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

A. public int

B. Long

C. void

D. private String

B. 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 incompat- ible 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.

30.Fill in the blank: A --------- variable is always available to all instances of the class.

A. public

B. local

C. static

D. instance

C. 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.

31.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;

A. 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.

32. 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

D. 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.

33.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.

C. 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.

33.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?

```
package useful;

public class MathHelper {

public static int roundValue(double d) {

// Implementation omitted

}

}
```

A.MathHelper:roundValue(5.92)

B.MathHelper.roundValue(3.1)

C.roundValue(4.1)

D.useful.MathHelper.roundValue(65.3)

B. 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.

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

```
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])));

}

}
```

A. None

B. One

C. Two

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

C. 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.

35.Fill in the blanks:----------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

D. 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 con- structor 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.

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

void run(String government)

A. Classes in other packages

B. Classes in the same package

C. Subclasses in a different package

D. All classes

B. 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.

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

```
package shield;

public class Protect {

private String material;

protected int strength;

public int getStrength() {
```

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

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.

A. 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.

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

```
package farm;
public class Coop {
public final int static getNumberOfChickens() {
```

```
// INSERT CODE HERE
}
}
```

A. return 3.0;

B. return 5L;

C. return 10;

D. None of the above.

D. 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.

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

```
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();
}
}
```

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.

C. The code contains a compilation problem in regard to the contents instance vari- able. 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 incom- patible, 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.

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

```
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());
}
```

}

A.static import clothes.Store.getClothes;

B.import clothes.Store.*;

C.import static clothes.Store.getClothes;

D.import static clothes.Store;

C. 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.

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

A.private

B.default

C.protected

D.None of the above

D. 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.

42.How many lines of the following program contain compilation errors?

```
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);
}
}
```

A.None

B.One

C.Two

D.Three

B. 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 state- ment 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.

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

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

A. return new Integer(3);

B. return new Byte((byte)6);

C. return 5L;

D. return new Short(4).longValue();

B. 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 con- structor 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.

44.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

C. 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.

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

```
package work;

public class Week {

private static final String monday;

String tuesday;

final static wednesday = 3;

final protected int thursday = 4;

}
```

A. One

B. Two

C. Three

D. The code will not compile regardless of the number of lines removed.

C. The declaration of monday does not compile, because the value of a static final vari- able 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 not compile because the final modifier cannot appear before the access modifier. For these reasons, Option C is the correct answer.

46.What is the output of the following application?

```
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
}
}
```

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.

D. 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 actu- ally 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.

48.Fill in the blanks: 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

A. 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.

49.What is the output of the following application?

```
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);
}
}
```

A.3

B.4

C.7

D.The code does not compile.

A. 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.

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

```
public class Drink {

public static void water() {}

public void get() {

// INSERT CODE HERE

}

}
```

A.water();

B.this.Drink.water();

C.this.water();

D.Drink.water();

B. 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.

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

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

A.call(9,"me",10,"Al")

B.call(5)

C.call(2,"home","sweet")

D.call("answering","service")

C. 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.

52.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.

D. 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.

53.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.

A. 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.

54.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.

D. 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.

55.What is the output of the following application?

```
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;
}
}
```

A.5

B.6

C.8

D.The code does not compile.

D. Since a constructor call is not the first line of the RainForest() constructor, the com- piler 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.

56.What is the output of the following application?

```
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));
}
}
```

A.5

B.2

C.11

D.The code does not compile.

A. 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. There- fore, 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.

57.What is the output of the following application?

```
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
}
}
```

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.

C.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 vari- able type being passed to the getScore() method on line m2, and Option C is correct.

58.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

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

59.How many lines of the following program contain compilation errors?

```
package theater;
class Cinema {
private String name;
public Cinema(String name) {this.name = name;}
}
public class Movie extends Cinema {
public Movie(String movie) {}
public static void main(String[] showing) {
System.out.print(new Movie("Another Trilogy").name);
```

}

}

A.None

B.One

C.Two

D.Three

C. The code does not compile, so Option A is incorrect. This code does not compile for two reasons. First, the name variable is marked private in the Cinema class, which means it cannot be accessed directly in the Movie class. Next, the Movie class defines a constructor that is missing an explicit super() statement. Since Cinema does not include a no-argument constructor, the no-argument super() cannot be inserted automatically by the compiler without a compilation error. For these two reasons, the code does not compile, and Option C is the correct answer.

60.Which modifier can be applied to an abstract interface method?

A. protected

B. static

C. final

D. public

D. All abstract interface methods are implicitly public, making Option D the correct answer. Option A is incorrect because protected conflicts with the implicit public modi- fier. Since static methods must have a body and abstract methods cannot have a body, Option B is incorrect. Finally, Option C is incorrect. A method, whether it be in an inter- face or a class, cannot be declared both final and abstract, as doing so would prevent it from ever being implemented.

61.What is the output of the following application?

```
package radio;
public class Song {
public void playMusic() {
System.out.print("Play!");
}
private static int playMusic() {
System.out.print("Music!");
}
public static void main(String[] tracks) {
new Song().playMusic();
}
}
```

A. Play!

B. Music!

C. The code does not compile.

D. The code compiles but the answer cannot be determined until runtime.

C. A class cannot contain two methods with the same method signature, even if one is static and the other is not. Therefore, the code does not compile because the two declara- tions of playMusic() conflict with one another, making Option C the correct answer.

62.Given the class declaration below, which value cannot be inserted into the blank line that would allow the code to compile?

```
package mammal;
interface Pet {}
public class Canine implements Pet {
public----------getDoggy() {
```

```
return this;
}
}
```

A.Class

B.Pet

C.Canine

D.Object

A. Recall that this refers to an instance of the current class. Therefore, any superclass of Canine can be used as a return type of the method, including Canine itself, making Option C an incorrect answer. Option B is also incorrect because Canine implements the Pet interface. An instance of a class can be assigned to any interface reference that it inherits. Option D is incorrect because Object is the superclass of instances in Java. Finally, Option A is the correct answer. Canine cannot be returned as an instance of Class because it does not inherit Class.

63.Which modifier can be applied to an interface method?

A. protected

B. static

C. private

D. final

B. Interface methods are implicitly public, making Option A and C incorrect. Interface methods can also not be declared final, whether they are static, default, or abstract methods, making Option D incorrect. Option B is the correct answer because an interface method can be declared static.

64.What is the output of the following application?

```
package track;
interface Run {
default void walk() { System.out.print("Walking and running!");
}
}
interface Jog {
default void walk() { System.out.print("Walking and jogging!");
}
}
public class Sprint implements Run, Jog
{
public void walk()
{
System.out.print("Sprinting!");
}
public static void main() {
new Sprint().walk();
}
}
```

A. Walking and running!

B. Walking and jogging!

C. Sprinting!

D. The code does not compile.

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 com- piles without issue, and Option C is correct.

65.Which of the following statements about interfaces is not true?

A. An interface can extend another interface.

B. An interface can implement another interface.

C. A class can implement two interfaces.

D. A class can extend another class.

B. Interfaces can extend other interfaces, making Option A incorrect. On the other hand, an interface cannot implement another interface, making Option B the correct answer. A class can implement any number of interfaces, making Option C incorrect. Finally, a class can extend another class, making Option D incorrect.

66.What is the output of the following application?

```
package transport;
class Ship {
protected int weight = 3;
private int height = 5;
public int getWeight() { return weight; }
public int getHeight() { return height; }
}
public class Rocket extends Ship {
public int weight = 2;
public int height = 4;
public void printDetails() {
System.out.print(super.getWeight()+","+super.height);
}
public static final void main(String[] fuel) {
```

```
new Rocket().printDetails();
}
}
```

A.2,5

B.3,4

C.3,5

D.The code does not compile.

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.

67.Fill in the blanks: Excluding default and static methods, a(n)---------- can contain both abstract and concrete methods, while a(n)----------contains only abstract methods.

A. concrete class, abstract class

B. concrete class, interface

C. interface, abstract class

D. abstract class, interface

68.Which statement about the following class is correct?

```
package shapes;
abstract class Triangle {
abstract String getDescription();
}
class RightTriangle extends Triangle {
protected String getDescription() { return "rt"; } // g1
```

```
}
public abstract class IsoscelesRightTriangle extends RightTriangle{//g2
public String getDescription() {return "irt"; }
public static void main(String[] edges) {
final Triangle shape = new IsoscelesRightTriangle(); // g3
System.out.print(shape.getDescription());
}
}
```

A. The code does not compile due to line g1.

B. The code does not compile due to line g2.

C. The code does not compile due to line g3.

D. The code compiles and runs without issue.

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.

69.Given that Short and Integer extend Number, what type can be used to fill in the blank in the class below to allow it to compile?

```
package band;
interface Horn
{
public Integer play();
}
abstract class Woodwind
{
```

```
 public Short play()
 {
return 3;
}
}
public final class Sphone extends Woodwind implements Horn {
public------------play()
{
 return null;
}
}
```

A. Integer

B. Short

C. Number

D. None of the above

D. The play() method is overridden in Sphone 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 Sphone class cannot compile regardless of its implementation of play(), making Option D the correct answer.

70.Fill in the blanks:

A class-----------an interface, while a class------------ an abstract class.

A. extends, implements

B. extends, extends

C. implements, extends

D. implements, implements

C. A class can implement an interface, not extend it. Alternatively, a class extends an abstract class. Therefore, Option C is the correct answer.

71.What is the output of the following application?

```
package paper;
abstract class Book {
protected static String material = "papyrus";
public Book() {}
public Book(String material) {this.material = material;}
}
public class Encyclopedia extends Book {
public static String material = "cellulose";
public Encyclopedia() {super();}
public String getMaterial() {return super.material;}
public static void main(String[] pages) {
System.out.print(new Encyclopedia().getMaterial());
}
}
```

A. papyrus

B. cellulose

C. The code does not compile.

D. The code compiles but throws an exception at runtime.

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 vari- ables, 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 con- structor Book(String) is not used in the Encyclopedia class.

72.Which of the following modifiers can be applied to an abstract method?

A. final

B. private

C. default

D. protected

D. An abstract method cannot include the final or private method. If a class contained either of these modifiers, then no concrete subclass would ever be able to override them with an implementation. For these reasons, Options A and B are incorrect. Option C is also incorrect because the default keyword applies to concrete interface methods, not abstract methods. Finally, Option D is correct. The protected, package-private, and public access modifiers can each be applied to abstract methods.

73.Given Of the following four modifiers, choose the one that is not implicitly applied to all interface variables.

A. final

B. abstract

C. static

D. public

B. Interface variables are implicitly public, static, and final. Variables cannot be declared as abstract in interfaces, nor in classes.

74.What is the output of the following application?

```
package race;

abstract class Car {

static { System.out.print("1"); }
```

```
public Car(String name) {
super();
System.out.print("2");
}
{ System.out.print("3"); }
}
public class BlueCar extends Car {
{
System.out.print("4");
}
public BlueCar()
{
super("blue");
System.out.print("5");
}
public static void main(String[] gears) {
new BlueCar();
}
}
```

A. 23451

B. 12354

C. 13245

D. The code does not compile.

C. The class is loaded first, with the static initialization block called and 1 is output- ted 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. The result is that 13245 is printed, mak- ing Option C the correct answer.

74.Fill in the blank: Overloaded and overridden methods always ha .

A. the same parameter list

B. different return types

C. the same method name

D. covariant return types

C. Overloaded methods share the same name but a different list of parameters and an optionally different return type, while overridden methods share the exact same name, list of parameters, and return type. For both of these, the one commonality is that they share the same method name, making Option C the correct answer.

75.Fill in the blank: A(n)---------- is the first non-abstract subclass that is required to implement all of the inherited abstract methods.

A. abstract class

B. abstraction

C. concrete class

D. interface

C. A concrete class is the first non-abstract subclass that extends an abstract class and implements any inherited interfaces. It is required to implement all inherited abstract meth- ods, making Option C the correct answer.

76.Fill in the blanks: An interface----------another interface, while a class----------another class.

A. implements, extends

B. extends, extends

C. implements, implements

D. extends, implements

B. An interface can only extend another interface, while a class can only extend another class. A class can also implement an interface, although that comparison is not part of the question text. Therefore, Option B is the correct answer.

77.Which of the following statements about no-argument constructors is correct?

A.If a parent class does not include a no-argument constructor, a child class cannot declare one.

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.

C.If a parent class contains a no-argument constructor, a child class must contain a no-argument constructor.

D.If a parent class contains a no-argument constructor, a child class must contain at least one constructor.

B. If a parent class does not include a no-argument constructor, a child class can still explicitly declare one; it just has to call an appropriate parent constructor with super(), making Option A incorrect. If a parent class does not include a no-argument constructor, the child class must explicitly declare a constructor, since the compiler will not be able to insert the default no-argument constructor, making Option B correct. Option C is incorrect because a parent class can have a no-argument constructor, while its subclasses do not. If Option C was true, then all classes would be required to have no-argument constructors since they all extend java.lang.Object, which has a no-argument constructor. Option D is also incorrect. The default no-argument constructor can be inserted into any class that directly extends a class that has a no-argument constructor. Therefore, no constructors in the subclass are required.

78.What is the result of compiling and executing the following application?

```
package mind;
public class Remember {
public static void think() throws Exception { // k1 try
{
throw new Exception();
}
}
public static void main(String... ideas) throws Exception {
think();
}
}
```

A. The code compiles and runs without printing anything.

B. The code compiles but a stack trace is printed at runtime.

C. The code does not compile because of line k1.

D. The code does not compile for another reason.

D. A try block must include either a catch or finally block, or both. The think() method declares a try block but neither additional block. For this reason, the code does not compile, and Option D is the correct answer. The rest of the lines compile without issue, including k1.

79.Choose the answer that lists the keywords in the order that they would be used together.

A. catch, try, finally

B. try, catch, finally

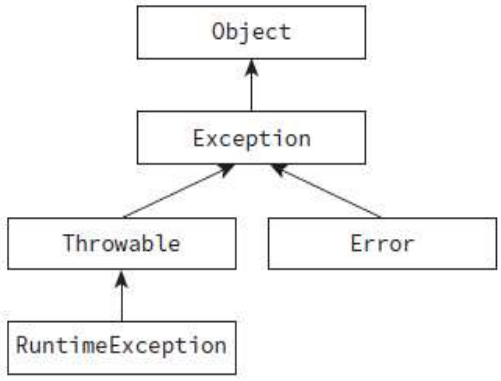
C. finally, catch, try

D. try, finally, catch

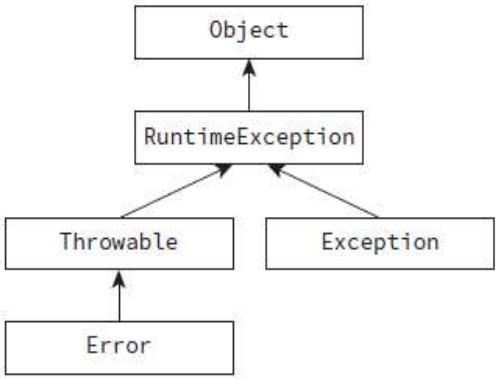
B. The correct order of blocks is try, catch, and finally, making Option B the correct answer.

80.Which of the following diagrams of java.lang classes shows the inheritance model properly?

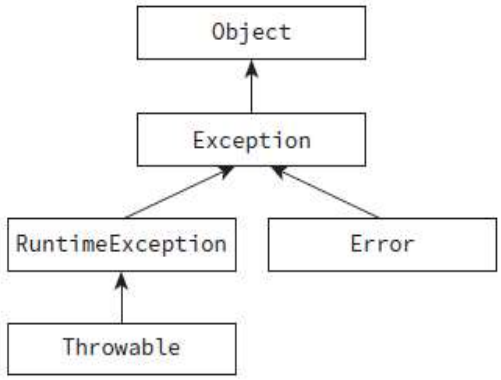
A.



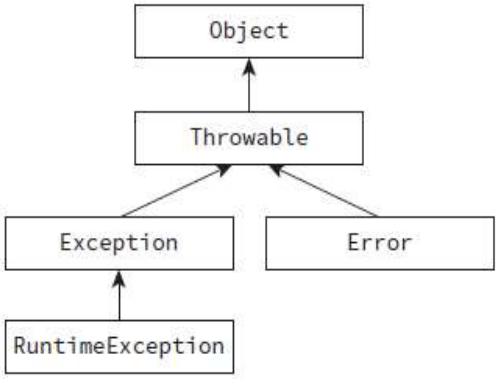
B.



C.



D.



D. Option D is the correct model. The class RuntimeException extends Exception, and both Exception and Error extend Throwable. Finally, like all Java classes, they all inherit from Object. Notice that Error does not extend Exception, even though we often refer to these generally as exceptions.

81.Which of the following Throwable types is it recommended not to catch in a Java application?

A.Error

B.CheckedException

C.Exception

D.RuntimeException

A. While Exception and RuntimeException are commonly caught in Java applications, it is recommended Error not be caught. An Error often indicates a failure of the JVM which cannot be recovered from. For this reason, Option A is correct, and Options C and D are incorrect. Option B is not a class defined in the Java API; therefore, it is also incorrect.

82.What is the output of the following application?

```
package game;
public class Baseball {
public static void main(String... teams) {
try {
int score = 1;
System.out.print(score++);
}
catch (Throwable t) {
System.out.print(score++);
}
finally {
System.out.print(score++);
}
System.out.print(score++);
}
}
```

A.123

B.124

C.12

D.None of the above

D. The application does not compile because score is defined only within the try block. The other three places it is referenced, in the catch

block, in the finally block, and out- side the try-catch-finally block at the end, are not in scope for this variable and each does not compile. Therefore, the correct answer is Option D.

83.Which of the following is a checked exception?

A. ClassCastException

B. IOException

C. ArrayIndexOutOfBoundsException

D. IllegalArgumentException

B.ClassCastException, ArrayIndexOutOfBoundsException, and IllegalArgumentException are unchecked exceptions and can be thrown at any time. IOException is a checked exception that must be handled or declared when used, mak- ing Option B the correct answer.

84.Fill in the blanks: The----------keyword is used in method declarations, while the----------keyword is used to throw an exception to the surrounding process.

A. throws, throw

B. catch, throw

C. throw, throws

D. throws, catch

A. The throws keyword is used in method declarations, while the throw keyword is used to throw an exception to the surrounding process, making Option A the correct answer. The catch keyword is used to handle exceptions, not to create them or in the declaration of a method.

85.If a try statement has catch blocks for both Exception and IOException, then which of the following statements is correct?

A.The catch block for Exception must appear before the catch block for IOException.

B.The catch block for IOException must appear before the catch block for Exception.

C.The catch blocks for these two exception types can be declared in any order.

D.A try statement cannot be declared with these two catch block types because they are incompatible.

B. IOException is a subclass of Exception, so it must appear first in any related catch blocks. If Exception was to appear before IOException, then the IOException block would be considered unreachable code because any thrown IOException is already han- dled by the Exception catch block. For this reason, Option B is correct.

86.What is the output of the following application?

```
package game;
public class Football {
public static void main(String officials[]) {
try {
System.out.print('A');
throw new RuntimeException("Out of bounds!");
}
catch (ArrayIndexOutOfBoundsException aioobe) {
System.out.print('B');
throw t;
}
finally { System.out.print('C');
}
}
}
```

A. ABC

B. ABC, followed by a stack trace for a RuntimeException

C. AC, followed by a stack trace for a RuntimeException

D. None of the above

C. The application first enters the try block and outputs A. It then throws a RuntimeException, but the exception is not caught by the catch block since RuntimeException is not a subclass of ArrayIndexOutOfBoundsException (it is a superclass). Next, the finally block is called and C is output. Finally, the RuntimeException is thrown by the main() method and a stack trace is printed. For these reasons, Option C is correct.

87.What is the result of compiling and running the following application?

```
package castles;
public class Fortress {
public void openDrawbridge() throws Exception { // p1 try {
throw new Exception("Circle");
}
catch (Exception e) {
System.out.print("Opening!");
}
finally {
System.out.print("Walls"); // p2
}
}
public static void main(String[] moat) {
new Fortress().openDrawbridge(); // p3
}
}
```

A. The code does not compile because of line p1.

B. The code does not compile because of line p2.

C. The code does not compile because of line p3.

D. The code compiles, but a stack trace is printed at runtime.

C. The application does not compile, so Option D is incorrect. The openDrawbridge() method compiles without issue, so Options A and B are incorrect. The issue here is how the openDrawbridge() method is called from within the main() method on line p3. The openDrawbridge() method declares the checked exception, Exception, but the main() method from which it is called does not handle or declare the exception. In order for this code to compile, the main() method would have to have a try-catch statement around line p3 that properly handles the checked exception, or the main() would have to be updated to declare a compatible checked exception. For these reasons, line p3 does not compile, and Option C is the correct answer.

88.Which of the following exception types must be handled or declared by the method in which they are thrown?

A. NullPointerException

B. Exception

C. RuntimeException

D. ArithmeticException

B.NullPointerException and ArithmeticException both extend RuntimeException, which are unchecked exceptions and not required to be handled or declared in the method in which they are thrown. On the other hand, Exception is a checked exception and must be handled or declared by the method in which it is thrown. Therefore, Option B is the correct answer.

89.What is the output of the following application?

```
package game;

public class BasketBall {

public static void main(String[] dribble) {
```

```
try
{
System.out.print(1);
throw new ClassCastException();
}
catch (ArrayIndexOutOfBoundsException ex)
{
System.out.print(2);
}
catch (Throwable ex)
{
System.out.print(3);
}
finally
{
System.out.print(4);
}
System.out.print(5);
}
}
```

A.1345

B.1235

C.The code does not compile.

D.The code compiles but throws an exception at runtime.

A. The code compiles and runs without issues, so Options C and D are incorrect. The try block throws a ClassCastException. Since ClassCastException is not a subclass of ArrayIndexOutOfBoundsException, the first catch block is skipped. For the second catch block,

ClassCastException is a subclass of Throwable, so that block is executed. Afterward, the finally block is executed and then control returns to the main() method with no exception being thrown. The result is that 1345 is printed, making Option A the correct answer.

90.Which of the following statements about a finally block is true?

A.Every line of the finally block is guaranteed to be executed.

B.The finally block is executed only if the related catch block is also executed.

C.The finally statement requires brackets {}.

D.The finally block cannot throw an exception.

C. A finally block can throw an exception, in which case not every line of the finally block would be executed. For this reason, Options A and D are incorrect. Option B is also incorrect The finally block is called regardless of whether or not the related catch block is executed. Option C is the correct answer. Unlike an if-then statement, which can take a single statement, a finally statement requires brackets {}.

91.Given that FileNotFoundException is a subclass of IOException, what is the output of the following application?

```
package office;

import java.io.*;

public class Printer {

public void print()

{

try

{

throw new FileNotFoundException();

}

catch (IOException exception)
```

```
{
System.out.print("Z");
}
catch (FileNotFoundException enfe)
{
System.out.print("X");
}
finally
{
System.out.print("Y");
}
}
public static void main(String... ink)
{
new Printer().print();
}
}
```

A. XY

B. ZY

C. The code does not compile.

D. The code compiles but a stack trace is printed at runtime.

C. The code does not compile because the catch blocks are used in the wrong order. Since IOException is a superclass of FileNotFoundException, the FileNotFoundException is considered unreachable code. For this reason, the code does not compile, and Option C is correct.

92.Which keywords are required with a try statement?

I.catch

II.finalize

III.finally

A.I only

B.II only

C.I or III, or both

D.None of these statements are required with a try statement.

C. A try statement requires a catch or a finally block. Without one of them, the code will not compile; therefore, Option D is incorrect. A try statement can also be used with both a catch and finally block, making Option C the correct answer. Note that finalize is not a keyword, but a method inherited from java.lang.Object.

93.Which statement about the role of exceptions in Java is incorrect?

A.Exceptions are often used when things "go wrong" or deviate from the expected path.

B.An application that throws an exception will terminate.

C.Some exceptions can be avoided programmatically.

D.An application that can properly handle its exception may recover from unexpected problems.

B. Option A is a true statement about exceptions and when they are often applied. Option B is the false statement and the correct answer. An application that throws an exception can choose to handle the exception and avoid termination. Option C is also a true statement. For example, a NullPointerException can be avoided on a null object by testing whether or not the object is null before attempting to use it. Option D is also a correct statement. Attempting to recover from unexpected problems is an important aspect of proper exception handling.

94.What is the output of the following application?

```
package harbor;

class CapsizedException extends Exception {}

class Transport {

public int travel() throws CapsizedException { return 2; };

}

public class Boat {

public int travel() throws Exception { return 4; }; // j1 public static
void main(String... distance) throws Exception

{

try

{

System.out.print(new Boat().travel());

}

catch (Exception e) ( System.out.print(8);)

}

}
```

A.4

B.8

C.The code does not compile due to line j1.

D.The code does not compile for another reason.

D. The code does not compile because the catch block uses parentheses () instead of brackets {}, making Option D the correct answer. Note that Boat does not extend Transport, so while the override on line j1 appears to be invalid since Exception is a broader checked exception than CapsizedException, that code compiles without issue. If the catch block was fixed, the code would output 4, making Option A the correct answer.

95.Which import statement is required to be declared in order to use the Exception, RuntimeException, and Throwable classes in an application?

A. import java.exception.*;

B. import java.util.exception.*;

C. import java.lang.*;

D. None of the above

D. All three of those classes belong to the java.lang package, so Option C seems like the correct answer. The Java compiler, though, includes java.lang by default, so no import statement is actually required to use those three classes, making Option D the correct answer.

96.Fill in the blanks: A program must handle or declare----------but should never handle---------------.

A.java.lang.Error, unchecked exceptions

B.checked exceptions, java.lang.Error

C.java.lang.Throwable, java.lang.Error

D.unchecked exceptions, java.lang.Exception

B. Checked exceptions must be handled or declared or the program will not compile, while unchecked exceptions can be optionally handled. On the other hand, java.lang.Error should never be handled by the application because it often indicates an unrecoverable state in the JVM, such as running out of memory. For these reasons, Option B is the correct answer.

97.If an exception matches two or more catch blocks, which catch block is executed?

A.The first one that matches is executed.

B.The last one that matches is executed.

C.All matched blocks are executed.

D.It is not possible to write code like this.

A. If an exception matches multiple catch blocks, the first one that it encounters will be the only one executed, making Option A correct, and Options B and C incorrect. Option D is also incorrect. It is possible to write two consecutive catch blocks that can catch the same exception, with the first type being a subclass of the second. In this scenario, an exception thrown of the first type would match both catch blocks, but only the first catch block would be executed, since it is the more specific match.

98.In the following application, the value of list has been omitted. Assuming the code compiles without issue, which one of the following is not a possible output of executing this class?

```
package checkboard;

public class Attendance {

private Boolean[] list = // value omitted

public int printTodaysCount() {

int count=0;

for(int i=0; i<10; i++) {

if(list[i])

++count;

}

return count;

}

public static void main(String[] roster)

{

new Attendance().printTodaysCount();

}

}
```

A.A stack trace for NullPointerException is printed.

B.A stack trace for ArrayIndexOutOfBoundsException is printed.

C.A stack trace for ClassCastException is printed.

D.None of the above

D. A NullPointerException can be thrown if the value of list is null. Likewise, an ArrayIndexOutOfBoundsException can be thrown if the value of list is an array with fewer than 10 elements. Finally, a ClassCastException can be thrown if list is assigned an object that is not of type Boolean[]. For example, the assignment

list = (Boolean[]) new Object() will compile without issue but throws a ClassCastException at runtime. Therefore, the first three options are possible, making Option D the correct answer.

99.Fill in the blanks: A----------occurs when a program recourses too deeply into an infinite loop, while a(n)----------occurs when a reference to a nonexistent object is acted upon.

A.NoClassDefFoundError, StackOverflowError

B.StackOverflowError, NullPointerException

C.ClassCastException, IllegalArgumentException

D.StackOverflowError, IllegalArgumentException

B. A StackOverflowError occurs when a program recurses too deeply into an infinite loop. It is considered an error because the JVM often runs out of memory and cannot recover. A NullPointerException occurs when an instance method or variable on a null reference is used. For these reasons, Option B is correct. A NoClassDefFoundError occurs when code available at compile time is not available at runtime. A ClassCastException occurs when an object is cast to an incompatible reference type. Finally, an IllegalArgumentException occurs when invalid parameters are sent to a method.

100.Fill in the blanks: A try statement has----------finally block(s) and---------- catch blocks.

A. zero or one, zero or more

B. one, one or more

C. zero or one, zero or one

D. one or more, zero or one

A. A try statement is not required to have a finally block, but if it does, there can be at most one. Furthermore, a try statement can have any number of catch blocks or none at all. For these reasons, Option A is the correct answer.

101.What is the output of the following application?

```
package pond;
abstract class Duck
{
protected int count;
public abstract int getDuckies();
}
public class Ducklings extends Duck {
private int age;
public Ducklings(int age) {
this.age = age;
}
public int getDuckies(){
return this.age/count;
}
public static void main(String[] pondInfo)
{
Duck itQuacks = new Ducklings(5);
System.out.print(itQuacks.getDuckies());
}
}
```

A. 0

B. 5

C. The code does not compile.

D. The code compiles but throws an exception at runtime.

D. The code compiles without issue, so Option C is incorrect. The key here is noticing that count, an instance variable, is initialized with a value of 0. The getDuckies() method ends up computing 5/0, which leads to an unchecked ArithmeticException at runtime, making Option D the correct answer.

102. Given a try statement, if both the catch block and the finally block each throw an exception, what does the caller see?

A.The exception from the catch block

B.The exception from the finally block

C.Both the exception from the catch block and the exception from the finally block

D.None of the above

B. If both the catch and finally blocks throw an exception, the one from the finally block is propagated to the caller, with the one from the catch block being dropped, making Option B the correct answer. Note that Option C is incorrect due to the fact that only one exception can be thrown to the caller.

103.Which of the following classes will handle all types in a catch block?

A.Exception

B.Error

C.Throwable

D.RuntimeException

C. All exceptions in Java inherit from Throwable, making Option C the correct answer. Note that Error and Exception extend Throwable, and RuntimeException extends Exception.

104.If a try statement has catch blocks for both ClassCastException and RuntimeException, then which of the following statements is correct?

A.The catch block for ClassCastException must appear before the catch block for RuntimeException.

B.The catch block for RuntimeException must appear before the catch block for ClassCastException.

C.The catch blocks for these two exception types can be declared in any order.

D.A try statement cannot be declared with these two catch block types because they are incompatible.

A. ClassCastException is a subclass of RuntimeException, so it must appear first in any related catch blocks. If RuntimeException was to appear before ClassCastException, then the ClassCastException block would be considered unreachable code, since any thrown ClassCastException is already handled by the RuntimeException catch block. For this reason, Option A is correct.

105.Which of the following is the best scenario to use an exception?

A.The computer caught fire.

B.The code does not compile.

C.A caller passes invalid data to a method.

D.A method finishes sooner than expected.

C. Option A is incorrect. You should probably seek help if the computer is on fire! Option B is incorrect because code that does not compile cannot run and therefore cannot throw any exceptions. Option C is the best answer, since an IllegalArgumentException can be used to alert a caller of missing or invalid data. Option D is incorrect; finishing sooner is rarely considered a problem.

106.Which statement about the following exception statement is correct?

throw new NullPointerException();

A.The code where this is called must include a try-catch block that handles this exception.

B.The method where this is called must declare a compatible exception.

C.This exception cannot be handled.

D.This exception can be handled with a try-catch block or ignored altogether by the surrounding method.

D. A NullPointerException is an unchecked exception. While it can be handled by the surrounding method, either through a try-catch block or included in the method declara- tion, these are optional. For this reason, Option D is correct.

107.If a try statement has catch blocks for both IllegalArgumentException and ClassCastException, then which of the following statements is correct?

A.The catch block for IllegalArgumentException must appear before the catch block for ClassCastException.

B.The catch block for ClassCastException must appear before the catch block for IllegalArgumentException.

C.The catch blocks for these two exception types can be declared in any order.

D.A try statement cannot be declared with these two catch block types because they are incompatible.

C.Both IllegalArgumentException and ClassCastException inherit RuntimeException, but neither is a subclass of the other. For this reason, they can be listed in either order, making Option C the correct statement.

108.What is the output of the following application?

```
package broken;
class Problem implements RuntimeException
{}
public class BiggerProblem extends Problem {
public static void main(String uhOh[]) {
try {
throw new BiggerProblem();
}
catch (BiggerProblem re) {
System.out.print("Problem?");
}
catch (Problem e) {
System.out.print("Handled");
}
finally {
System.out.print("Fixed!");
}
}
}
```

A.Problem?Fixed!

B.Handled.Fixed!

C.Problem?Handled.Fixed!

D.The code does not compile.

D. The class RuntimeException is not an interface and it cannot be implemented. For this reason, the Problem class does not compile, and Option D is the correct answer. Note that this is the only compilation problem in the application. If implements was changed to extends, the code would compile and Problem?Fixed! would be printed, making Option A the correct answer.

109.Given an application that hosts a website, which of the following would most likely result in a java.lang.Error being thrown?

A.Two users try to register an account at the same time.

B.The application temporarily loses connection to the network.

C.A user enters their password incorrectly.

D.The application runs out of memory.

D. A Java application tends to only throw an Error when the application has entered a final, unrecoverable state. Options A and C are incorrect. These types of errors are common and expected in most software applications, and should not cause the application to terminate. Option B uses the word temporarily, meaning the network connection will come back. In this case, a regular exception could be used to try to recover from this state. Option D is the cor- rect answer because running out of memory is usually unrecoverable in Java.

110.What is the output of the following application?

```
package bed;
public class Sleep {
public static void snore() {
try {
String sheep[] = new String[3];
System.out.print(sheep[3]);
}
catch (RuntimeException e) {
System.out.print("Awake!");
}
finally {
throw new Exception(); // x1
}
}
```

```
public static void main(String... sheep) {
// x2 new Sleep().snore(); // x3
}
}
```

A. Awake!, followed by a stack trace

B. The code does not compile because of line x1.

C. The code does not compile because of line x2.

D. The code does not compile because of line x3.

B. The finally block of the snore() method throws a new checked exception on line x1, but there is no try-catch block around it to handle it, nor does the snore() method declare any checked exceptions. For these reasons, line x1 does not compile, and Option B is the correct answer. The rest of the lines of code compile without issue, even line x3 where a static method is being accessed using an instance reference. Note that the code inside the try block, if it ran, would produce an ArrayIndexOutOfBoundsException, which would be caught by the RuntimeException catch block, printing Awake!. What happens next would depend on how the finally block was corrected.