1.Which of the following variable types is not permitted in a switch statement?

A. String
B. double
C. int
D. char

B. A switch statement supports the primitive types byte, short, char, and int and the classes String, Character, Byte, Short, and Integer. It also supports enumerated types. Floating-point types like float and double are not supported, therefore Option B is the correct answer.

2.What is the value of tip after executing the following code snippet?

```
int meal = 5;
int tip = 2;
int total = meal + (meal>6 ? ++tip : --tip);
```

A. 1
B. 2
C. 3
D. 6

A. Remember that in ternary expressions, only one of the two right-most expressions are evaluated. Since meal>6 is false, --tip is evaluated and ++tip is skipped. The result is that tip is changed from 2 to 1, making Option A the correct answer. The value of total is 6, since the pre-increment operator was used on tip, although you did not need to know this to solve the question.

3.What is the output of the following application?

```
package registration;
public class NameCheck
{
public static void main(String... data)
{
String john = "john";
String jon = new String(john);
System.out.print((john==jon)+" "+(john.equals(jon)));
}
}
```

A. true true
B. true false
C. false true
D. false false

C. The first assignment creates a new String "john" object. The second line explicitly uses the new keyword, meaning a new String object is created. Since these objects are not the same, the == test on them evaluates to false. The equals() test on them returns true because the values they refer to are equivalent. Therefore, the correct answer is C.

4.Which statement immediately exits a switch statement, skipping all remaining case or default branches?

A. exit
B. break
C. goto
D. continue

B. The break statement exits a switch statement, skipping all remaining branches, mak- ing Option B the correct answer. In Option A, exit is not a statement in Java. In Option C, goto is a reserved word but unused in Java. Finally, in Option D, continue is a statement but only used for loops.

5.What is the output of the following application?

```
package dinosaur;
public class Park
{
public final static void main(String... arguments)
{
int pterodactyl = 6;
long triceratops = 3;
if(pterodactyl % 3 >= 1)
triceratops++; triceratops--;
System.out.print(triceratops);
}
}
```

A. 2
B. 3
C. 4
D. The code does not compile.

A. The first step is to determine whether or not the if-then statement's expression is executed. The expression 6 % 3 evaluates to 0, since there is no remainder, and since 0 >= 1 is false, the expression triceratops++ is not called. Notice there are no brackets {} in the if-then statement. Despite the triceratops-- line being indented, it is not part of the if-then statement. Recall that Java does not use indentation to determine the beginning or end of a statement. Therefore, triceratops-- is always executed, result- ing in a value of 2 for triceratops and making Option A the correct answer.

6.What is the output of the following application?

```
package restaurant;
public class Pieces
{
public static void main(String[] info)
{
int flair = 15;
if(flair >= 15 && flair < 37)
{
 System.out.print("Not enough");
}
if(flair==37)
{
 System.out.print("Just right");
}
else
{
System.out.print("Too many");
}
}
}
```

A. Not enough
B. Just right
C. Too many
D. None of the above

E. For this question, it helps to notice that the second if-then statement is not connected to the first if-then statement, as there is no else joining them. When this code executes, the first if-then statement outputs Not enough since flair is >= 15 and < 37. The second if- then statement is then evaluated. Since flair is

not 37, the expression Too many is output- ted. Since two statements are outputted, Option D, none of the above, is the correct answer.

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

```
int hops = 0; int jumps = 0; jumps = hops++; if(jumps)
System.out.print("Jump!"); else
System.out.print("Hop!");
```

A. Jump!
B. Hop!
C. The code does not compile.
D. The code compiles but throws an exception at runtime.

C. The value of jumps and hops is unimportant because this code does not compile, mak- ing Option C the correct answer. Unlike some other programming languages, Java does not automatically convert integers to boolean values for use in if-then statements. The state- ment if(jumps) evaluates to if(0), and since 0 is not a boolean value, the code does not compile. Note that the value of the jumps variable is irrelevant in this example; no integer evaluates to a boolean value in Java.

8.Fill in the blanks: The operator increases the value of a variable by 1 and returns the new value, while the operator decreases the value of a variable by 1 and returns the original value.

A. pre-increment [++v], pre-decrement [--v]
B. pre-increment [++v], post-decrement [v--]
C. post-increment [v++], pre-decrement [--v]
D. post-increment [v++], post-decrement [v--]

B. Prefix operators modify the variable and evaluate to the new value, while postfix opera- tors modify the variable but return the original value. Therefore, Option B is the correct answer.

9.What is the output of the following application?

```
package jungle;
public class TheBigRace
{
public static void main(String[] in)
{
int tiger = 2;
short lion = 3;
long winner = lion+2*(tiger + lion);
```

```
System.out.print(winner);
}
}
```

A. 11
B. 13
C. 25
D. None of the above

B. For this problem, it helps to recognize that parentheses take precedence over the opera- tions outside the parentheses. Once we replace the variables with values, the expression becomes: 3+2*(2+3). We then calculate the value inside the parentheses to get 3+2*5. Since the multiplication operator has higher precedence than addition, we evaluate it first, result- ing in 3+10 = 13, making Option B the correct answer.

10.Given the following code snippet, assuming dayOfWeek is an int, what variable type of saturday is not permitted?

```
final saturday = 6;
switch(dayOfWeek)
{
default:
System.out.print("Another Weekday");
break;
case saturday:
System.out.print("Weekend!");
}
```

A. byte
B. long
C. int
D. None of the above

B. Any value that can be implicitly promoted to int will work for the case statement with an int input. Since switch statements do not support long values, and long cannot be converted to int without a possible loss of data, Option B is the correct answer.

11.What is the output of the following application?

```
package recreation;
public class Dancing
{
public static void main(String[] vars)
{
int leaders = 10 * (2 + (1 + 2 / 5);
int followers = leaders * 2;
System.out.print(leaders + followers < 10 ? "Too few" : "Too many");
}
}
```

A. Too few
B. Too many
C. The code does not compile.
D. The code compiles but throws a division by zero error at runtime.

C. While the code involves numerous operations, none of that matters for solving this problem. The key to solving it is to notice that the line that assigns the leaders variable has an uneven number of parentheses. Without balanced parentheses, the code will not com- pile, making Option C the correct answer.

12.What is the output of the following application?

```
package schedule;
public class PrintWeek
{
public static final void main(String[] days)
{
 System.out.print(5 + 6 + "7" + 8 + 9);
}
}
```

A. 56789
B. 11789
C. 11717
D. The code does not compile.

B. Remember that Java evaluates + from left to right. The first two values are both num- bers, so the + is evaluated as numeric addition, resulting in a reduction to 11 + "7" + 8 + 9. The next two terms, 11 + "7", are handled as string concatenation since one of the terms is a String. This allows us to reduce the expression to "117" + 8 + 9. Likewise, the final two terms are each evaluated one at a time with the

String on the left. Therefore, the final value is 11789, making Option B the correct answer.

13.Fill in the blanks: The operator is used to find the difference between two numbers, while the operator is used to find the remainder when one number is divided by another.

A. /, %
B. –, %
C. %, <
D. –, ||

B. The subtraction - operator is used to find the difference between two numbers, while the modulus % operator is used to find the remainder when one number is divided by another, making Option B the correct answer. The other options use operators that do not match this description.

14.What is the output of the following application?

```
package transporter;
public class Rematerialize
{
public static void main(String[] input)
{
int dog = 11;
int cat = 3;
int partA = dog / cat;
int partB = dog % cat;
int newDog = partB + partA * cat;
System.out.print(newDog);
}
}
```

A. 9
B. 11
C. 15
D. The code does not compile.

B. The code compiles without issue, making Option D incorrect. The focus of this ques- tion is showing how the division and modulus of two numbers can be used to reconstitute one of the original operands. In this example, partA is the integer division of the two num- bers. Since 3 does not divide 11 evenly, it is rounded down to 3. The variable partB is the remainder from the first expression, which is 2. The newDog variable is an expression that reconstitutes the original value for dog

using the division value and the remainder. Note that due to operator precedence, the multiplication * operation is evaluated before the addition + operation. The result is the original value of 11 for dog is outputted by this program.

15.What is the output of the following application?

```
package dessert;
public class IceCream {
public final static void main(String... args) {
int flavors = 30;
int eaten = 0; switch(flavors) {
case 30: eaten++;
case 40: eaten+=2;
default: eaten--;
}
System.out.print(eaten);
}
}
```

A. 1
B. 2
C. 3
D. The code does not compile.

B. The code compiles without issue, so Option D is incorrect. In this question's switch statement, there are no break statements. Once the matching case statement, 30, is reached, all remaining case statements will be executed. The variable eaten is increased by 1, then 2, then reduced by 1, resulting in a final value of 2, making Option B the correct answer.

16.What is the output of the following application?

```
package mode;
public class Transportation {
public static String travel(int distance) {
return distance<1000 ? "train" : 10;
}
public static void main(String[] answer) {
System.out.print(travel(500));
}
}
```

A. Train
B. 10
C. The code does not compile.
D. The code compiles but throws an exception at runtime.

C.Ternary operations require both right-hand expressions to be of compatible data types. In this example, the first right-hand expression of the outer ternary operation is of type String, while the second right-hand expression is of type int. Since these data types are incompatible, the code does not compile, and Option C is the correct answer.

17.Fill in the blanks: Given two non-null String objects with reference names apples and oranges, if apples oranges evaluates to true, then apples oranges must also evaluate to true.

A. ==, equals()
B. !=, equals()
C. equals(), ==
D. equals(), =!

A. For this question, remember that if two String objects evaluate to true using ==, then they are the same object. If they are the same String object, equals() will trivially return true. Option A correctly reflects this principle. Option B is incorrect as two String objects that are not the same may still be equivalent in terms of equals(). For example, apples == new String(apples) evaluates to false, but equals() will evaluate to true on these String objects. Likewise, Options C and D are also incorrect because two String objects that are equivalent in terms of equals() may be different objects.

18.How many 1s are outputted when the following application is compiled and run?

```
package city;
public class Road {
public static void main(String... in) {
int intersections = 100;
int streets = 200;
if (intersections < 150) {
System.out.print("1");
}
else if (streets && intersections > 1000) {
System.out.print("2");
}
```

```
if (streets < 500)
System.out.print("1");
else
System.out.print("2");
}
}
```

A. None
B. One
C. Two
D. The code does not compile.

D. The code does not compile, making Option D the correct answer. The reason the code does not compile is due to the test in the second if-then statement. The expression (streets && intersections > 1000) is invalid because streets is not a boolean expression and cannot be used as the left-hand side of the conjunctive logical && operator. The line of code is designed to resemble the corrected expression (streets > 1000 && intersections > 1000. Notice the fixed expression requires two relational > operators. If the second if-then statement was corrected, then the applica- tion would compile and produce two 1's, making Option C the correct answer.

19.Which statement about the logical operators & and && is true?

A. The & and && operators are interchangeable, always producing the same results at runtime.
B. The & operator always evaluates both operands, while the && operator may only evaluate the left operand.
C. Both expressions evaluate to true if either operand is true.
D. The & operator always evaluates both operands, while the && operator may only evaluate the right operand.

B. The & and && (AND) operators are not interchangeable, as the conjunctive & operator always evaluates both sides of the expression, while the conditional conjunctive && operator only evaluates the right-hand side of the expression if the left side is determined to be true. This is why conditional operators are often referred to as short-circuit operators, skipping the right-hand side expression at runtime. For these reasons, Option B is the correct answer. Note that Option C is an incorrect statement as well, since it describes disjunctive (OR) operators.

20.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.

22.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.

22.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.

23.What is the output of the following application?

```
package transporter;
public class TurtleVsHare {
public static void main(String[] arguments) {
int turtle = 10 * (2 + (3 + 2) / 5);
int hare = turtle < 5 ? 10 : 25;
System.out.print(turtle < hare ? "Hare wins!" : "Turtle wins!");
}
}
```

A. Hare wins!
B. Turtle wins!
C. The code does not compile.
D. The code compiles but throws a division by zero error at runtime.

B. The code compiles and runs without issue, making Options C and D incorrect. The key here is understanding operator precedence and applying the parentheses to override precedence correctly. The first expression is evaluated as follows: 10 * (2 + (3 + 2) / 5) → 10 * (2 + 5 / 5) → 10 * (2 + 1) → 10 * 3, with a final value of 30 for turtle. Since turtle is not less than 5, a value of 25 is assigned to hare. Since turtle is not less than hare, the last expression evaluates to Turtle wins!, which is outputted to the console, making Option B the correct answer.

24.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.

25.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.

26. Given the following code snippet, what is the value of movieRating after it is executed?

```
int characters = 5;
int story = 3;
double movieRating = characters <= 4 ? 3 : story>1 ? 2 : 1;
```

A. 2.0
B. 3.0
C. The code does not compile but would compile if parentheses were added.
D. None of the above

A. While parentheses are recommended for ternary operations, especially embedded ones, they are not required, so Option C is incorrect.. The first ternary operation evaluates characters <= 4 as false, so the second ternary operation is executed. Since story > 1 is true, the final value of movieRating is 2.0, making Option A the correct answer.