Chapter 2 – Basic

Answers

1. D

   A short contains 16 bits, and can contain negative or positive numbers.

2. B

   A boolean variable can only contain the values true and false, written as words with all lowercase letters.

3. C

   A byte contains 8 bits, and can contain negative or positive numbers.

4. A, B, C, D

   An identifier is a word used by a program to name a class, a method, a variable or a label. Identifiers can only begin with a letter (lowercase or uppercase), a $ sign or an underscore _ . They cannot begin with numbers, and their length is unlimited.

5. A, B, C, D, E

   An identifier is a word used by a program to name a class, a method, a variable or a label. Identifiers can only begin with a letter (lowercase or uppercase), a $ sign or an underscore _ . They cannot begin with numbers, and their length is unlimited.

6. B

   The number of positive values available for an int is one less than the number of negative values available.

   A int's range is from -2^{31} to $2^{31} - 1$.

7. B

   The number of positive values available for a short is one less than

the number of negative values available.

A short's range is from -2^{15} to $2^{15} - 1$.

8. B

   A char has only positive values. A char's range is from 0 to $2^{16} - 1$

9. A

   The % operator is a remainder operator. When you divide 14 by 4, the remainder left is 2.

10. B

    The % operator is a remainder operator. When you divide 4 by 14, the result is 0, and the remainder left is 4.

11. B, C, D

    Answers A and E are incorrect since they are trying to assign a boolean value to the variable "number", which is an int.

    Answers B, C and D are fine since they are comparing and assigning valid values.

12. C, D, E

    The + operator in Java is automatically overloaded in the Java language to be used as an arithmetic operator and as a concatenation operator used for Strings.

    Answer A is incorrect since when adding a String and an int the result is a String, and here the code is trying to place the result into an int variable.

    Answer B is incorrect with the "?" operator since it is comparing an int with a String to see if they are equal, and this is not a valid comparison.

Answers C and D are correct since they are comparing and assigning valid values.

Answer E is correct since a String variable and an int variable can be added together and placed into a String variable.

13. A

The ^ operator is an exclusive OR operator, also called an XOR operator. This operator compares the bits (0 or 1) of the two numbers one by one. A 0^1 and a 1^0 both result in a 1.

0^0 and a 1^1 both result in a 0.

14. B

The ^ operator is an exclusive OR operator, also called an XOR operator. This operator compares the bits (0 or 1) of the two numbers one by one. A 0^1 and a 1^0 both result in a 1. A 0^0 and a 1^1 both result in a 0.

15. A

The ^ operator is an exclusive OR operator, also called an XOR operator. This operator compares the bits (0 or 1) of the two numbers one by one. A 0^1 and a 1^0 both result in a 1. A 0^0 and a 1^1 both result in a 0.

When comparing the bit values of the numbers 2 and 2 with ^, each number only has the 2 bit spot turned on, and the ^ operator sees this as 0. So the result has no bits turned on, all are in the 0 position, so the resulting number is 0.

16. A

When performing calculations with the ++ operator within an equation, when the ++ comes after the variable first the equation is performed, then the variables with the ++ are increased.

In this example the following occurs in order:

c = a + b, that is c = 8 + 2, c = 10

a++, a = 8 + 1, a = 9

b++, b = 2 + 1, b = 3

17. A

When performing calculations with the ++ and -- operators within an equation, when the ++ or -- come after the variable first the equation is performed, then the variables with the ++ are increased by 1 and the variables with the – are decreased by 1.

In this example the following occurs in order:

c = a + b, that is c = 8 + 2, c = 10

a--, a = 8 - 1, a = 7

b++, b = 2 + 1, b = 3

18. C

In this example we do not know what the args sent are, but the comparison to check the args length is a valid comparison. With the "&&" operator, the second part of the comparison is only checked if the first part of the comparison is true. If the String " Israel" is appended to the initial StringBuffer then the StringBuffer would then contain the text "Shalom Israel", which is not equal to the String "Shalom" in its content, so the result is always false. So the program goes into the "else" section and the word "Israel" is printed.

19. B

When the >> operator is used on the number –1, the number always remains –1.

20. B

For the line "x=x>>32", since x was –1, it remains –1.

For the line "y=y>>>32", since y was a –1 it also remains a –1.

For the line "x=x>>>31", x was –1, and now it turns to be 1.

Then “y=y+x” so y=(-1) + (1) = 0.

21. A, C

    Answers A and C are correct since they both use valid comparisons and assignments.

    Answer B is incorrect since it is using the ”!” operator on an int, and that is not valid. The “!” operator can only be used with a boolean.

    Answer D is incorrect since when two integers are added they result in an int, and here the result is trying to be placed into a byte. A cast to a byte would be needed to make this correct, such as:

    ```
    b = (byte)b + 1
    ```

    Answer E is incorrect since with an “if” statement the value checked (within the ()) needs to be a boolean value, and here it is an int.

22. A

    Answer A is correct since it uses valid comparisons and assignments.

    Answer B is incorrect since it is using the “!” operator on an int, and that is not valid. The “!” operator can only be used with a boolean.
    Answer C is incorrect since <> is not a valid operator in the Java language, this would result in a compilation error.

    Answer D is incorrect since when two integers are added they result in an int. In this case b is of type byte and 1 is of type int. In order to complete the calculation an automatic casting of the value of b from byte into int is performed by the JVM. The result is being placed into a byte variable. A cast to a byte would be needed to make this correct, such as:

    ```
    b = (byte)(b + 1);
    ```

    Answer E is incorrect since with an “if” statement the value checked (within the ()) needs to be a boolean value, and here it is attempting

to assign a new value to the variable "num". This could be corrected with using the following instead:

if (num==1)

23. A

Answer A is correct since a "%" operator can work with integers and with floating point variables. When doing a calculation with the "%" operator, if the two operands are of different types, then the operand of the lower type will be converted to the type of the second operand (which holds a larger range of values.)
In the case here, the operand 4 is an int and the operand x is a float, and a float can hold more values than an int, so the int is converted to a float and then the calculation is made. The result will contain a floating point as well.

Answers B and C are incorrect since a boolean can only be assigned the values true or false, and not the values 0 or 1.

Answer D and E are incorrect since the content in the "if" statement's parenthesis are not valid. This needs to contain a valid boolean expression.

24. A

When performing calculations with the ++ and -- operators within an equation, when the ++ or -- come after the variable first the equation is performed, then the variables with the ++ are increased by 1 and the variables with the – are decreased by 1.

In this example the following occurs in order:
c = a + b, that is c = 4 + 5, c = 9
a--, a = 4 - 1, a = 3
b--, b = 5 - 1, b = 4

25. A, B, D

Answers A and B are correct since a String and an int can be added with the “+” operator and assigned to a String variable.

Answer C is incorrect since the result of adding a String and an int with the “+” operator cannot be placed into an int variable.

Answer D is correct because the “&&” operator is used and not the “&” operator. With the “&&” operator if the first operand is false, then the second operand is never checked. With the “&” operator both operands are always checked. Since the String “str” is null, if “str.length()” were to be performed on it a runtime error would occur. In D the first operand in the evaluation is false, so the second one is never evaluated.

26. A, C, E

Answer A is correct since when comparing two numbers of different types, the value in the type of the smaller range is converted to the type of the larger range for the comparison process. Here the int is converted to a float to do the comparison “if (numi==numf)”, so this comparison is valid.

Answer B is incorrect since the “if” statement is trying to compare an int with an Integer object, and this is not a valid comparison.

Answer C is correct since both “numl1” and “numl2” are variables that hold the same reference to the same Integer object, so the == comparison results in true. For objects, the == operator sees if the references of the two operands are the same, whereas for numbers, the == operator sees if the content of the two operands is the same.

Answer D is incorrect since the two Integer variables refer to two different object references.

Answer E is correct since when String objects are assigned their values with = “xxx” and not with the “new ” keyword, the compiler

sees that the second variable has the same literal value as the first variable, so it assigns the two variables to both be references to the same object.

27. B

    In this case since the String variables are assigned their values with the new keyword, the variables each refer to a different reference, so they are not equal.

28. A

    When using the “?” operator the two values after the “?” and between the “.” are automatically converted to be the same type, regardless of which one is the correct answer. The item with the smaller range type is converted to the same type as the item of the larger range type, so here the int 2 is converted to be a float with the value 2.0.

29. B

    In this case the two numbers on the right side of the “?” are of the same type, so no conversion of types takes place. The variable num is > 1000 so the first number after the “?” is the answer, which is 8.

30. D

    When the two byte variables are added together the result is an int, and here the code is trying to place them into a byte variable. A cast to a byte would be needed to fix this code.

31. B,E,F,G

    The result of adding two numbers together with the “”+ operator results in at least an int, so any type greater than an int can hold the sum as well.

32. A,C

    Each time the “index” and “counter” variables have the same value the inner loop is exited, so the answer will never contain two equal values, and the first value will never be more than the second value

for this reason as well. The only possible answers are where the first value is greater than the second value, and where the first value is less than 4 and the second value less than 12.

33. A,C,D

    In this case each time the index and counter variables have the same value the continue keyword causes the loop to skips the statements after the word “continue” and the inner loop continues from the top.

34. D

    This is an example of a simple “if … else” block, which checks the values of the variables to see which String to print.

35. D

    In the switch block the first case 2+2 results in the int 4, and that matches the value in the variable “j”, so the word “Haifa” is printed. Since there is no “break” statement the program automatically goes into the second case, even though its value 2 does not match “j”, and the word “Tel-Aviv” is printed as well. Then the word “break” appears, so the program leaves the switch block.

36. C,E

    Answer A is incorrect since the “while” block needs to evaluate to a boolean, and here there is just an int within the parenthesis, which is invalid for a “while” block.

    Answer B is incorrect since it declares the variable to be an int within the “while” parenthesis, and this is not valid.

    Answers C and E are correct since the use proper assignments and comparisons with variable values.

    Answer D is incorrect since again the value in the “while” block parentheses needs to evaluate to a boolean, and here an assignment is taking place. To fix this there is the need to put “while (counter==1)” with another = added.

37 B

The range of positive numbers is one less than the range of negative numbers.

38. C

The number –1 represented in bits contains all ones. When using the >> operator the number on the number –1, the result remains –1, so the text “Tel-Aviv” is printed. Since the >> operator works only on int values, the variable “x”, which is of type byte, will be automatically converted to type int before the >> is performed. The resulting value is of type int.

39. E

The result when using the >> operator is an int, and here the code is trying to assign the value to a byte, which will cause a compilation error.

40. A,D,E

When adding two numbers with the “+” operator the result is at least an int. Types larger than an int can hold the result as well.

41. E

The evaluation in the parenthesis of an “if” block needs to be a boolean, and here in line 4 it some type of number. Therefore the code would not compile.

42. D

In this class the method fly is overridden many times, each time receiving a different parameter type. There is no method that accepts as its parameter the type char, but the compiler sees that since a char can be turned into an int implicitly, this conversion occurs and the version of the method that accepts an int is used.

43. A,B,C

Answer s A and B do not compile since the code is trying to put a value with a type of a larger range (char and short) into a type of a smaller range, here a byte.

Answer C does not compile since when placing a float into a long a cast to a long is required. Although a long can hold 64 bits and a float can hold 32 bits, when converting from a to a long to a float if there are numbers after the decimal point they will be lost. Since data can be lost a cast is required.

Answers D and E compile fine since a variable with a type with a smaller range is being placed into a variable with a type of a larger range, and no information would be lost in the cast.

44. D

    In this “for” loop two variables (index and num) are being declared and initialized in the initialization part of the loop. Then based on the evaluation in the “for” loop, the text “shalom” is printed 11 times.

45. A,B,C

    The % operator gives the remainder of the first number divided by the second number as the result. The sign of the first operand is always the sign of the result.

46. A

    The % operator uses the division operator “/” in its calculation, and with the “/” operator if the code tries to divide by 0 an ArithmeticException occurs, so with the “%” operator it occurs as well.

47. B

    When using the >> operator on a positive number the number will divide its value by 2, but when using the >>> on a negative number, the result is a large positive number.

48. A

    If you use the >> operator after the number –1, the result will always

be –1.

49. B

    The char is considered a numeric type, and only allows positive values.

50. A

    The result of "b1>>1" is –2, and the result of "b1>>>1" is a large positive number. In this case both of these results are being cast to byte, so only the right-most 8 bits of the result are used, and the rest are cut off. Therefore the value in the variable b2 will remain the same as it was, -2, since the left most bits were all 1's anyway. On the other hand, the value in the variable b3 had other numbers in those left-most bits, which made the number to be a big positive number, whereas the right-most bits were identical to b2. Since only the right-most bits remain in this case, the values in the two variables are both –2, and are equal.

51. B

    The < operand only compares numeric values.

52. A

    To describe how the "instanceof" keyword works, take for example: "a instance of b".

    In order for this expression to return true, the relationship of "a" to "b" can be one of the following:
    "a" can be the same class or a subclass of "b"
    "a" can implement the interface of "b"
    "a" can refer to the array of "b"

53. A

    The operators | and & are overloaded in the Java language to work with both boolean values and with numbers.

54. A

The check in the “if” block if “sbf.length()>6” results in false, and since the && operator is used here and this first expression returned false, the next evaluation is never evaluated. Since the first statement returned false the “else” keyword is implemented.

55. C

The char variable “c” needs to be cast to a short in order to place it into the short variable “sh”, since in this assignment information could be lost.

56. A

The type of variable that a switch statement evaluates needs to be either an int or any whole number that can be converted implicitly into an int.

57. A

The number 22 is a decimal number and the number 022 is an octal number. 022 converted to a decimal number is the number 18, so here 22 – 18 = 4.

58. A

Identifiers are names that a programmer uses for variables. They are case sensitive and have no limit on the length. Of course a name that is too long would be hard to read, so it is not very practical.

59. C

When adding two numbers the result will be an int, and here the equation is trying to put the sum into a short variable, which holds less than an int. A cast to a short would be needed to fix this problem.

60. C

When adding two numbers the result will be an int, and here the equation is trying to put the sum, cast to a short, into a byte variable. A byte holds less than a short so this will not work. A cast to a byte instead of to a short would be needed to fix this code.

61. C

When adding two numbers the result will be an int, and here the equation is trying to put the sum into a byte variable, which holds less than an int. Casting the sum to a byte would be needed to fix this problem.

62. B,E,F

Java keywords are words that are built into the Java language, and have special meanings, and therefore cannot be used as identifiers.

63. A

A boolean type holds 1 bit, holding one of two values, true or false.

64. A,D,E

When adding two numbers with the + operator the variable holding the result has to be at least the size of an int, which holds 32 bits.

65. B

Although both short and char types hold 16 bits, the range of numbers that each one can hold is different. A short variable can hold both negative and positive values, and a char variable can hold only positive values.

66. D

A compilation error will occur at line 8, since the value in the “if” parenthesis does not evaluate to a boolean.

67. B

The number –1 is represented in bits as all 1’s. Using the >> operator on the number –1 always results in a –1.

68. A,C

The number numI1 is never assigned a new value, so it remains as –1. The number in numI2 will be a large positive value after the >>> is performed on it. The number in numB is assigned the value that is in

numI2 and converted to a byte, so the result is –1.

69. A,C

The number numI1 is never assigned a new value, so it remains as –1. The variable numI2 is an int, and gets assigned the result of "numI1>>>3", which is a positive number with a lot of digits, of type int. Then this result is cast to a byte, and that result is turned placed into the numB variable. Since the int value is being cast to a byte, the left-most 24 bits are cut off. The remaining right-mist 8 bits represent the number –1, which is the value placed into numB.

70. E

When adding two numbers with the "+" operator the compiler will change the two numbers to be of the same type before adding them. The number of the smaller type will be changed to be the same type as the number of the larger type. So here the int will be converted to a double and then the calculation will be made. The type of the sum would then be the same type as the operands, which is here a double.

71. A, C, D

Java keywords are words that are built into the Java language, and have special meanings, and therefore cannot be used as identifiers.

72. D

A compilation error would occur at line 7, since the result of adding two numbers is by default an int. A cast to a byte would be needed to fix this code.

73. C

In this class there are statements, but they are not within a method, so a compilation error will occur. In order to fix this all of these statements should be placed within a "main" method, and then the code will compile.

74. D,E

When adding two numbers, the result type needs to be at least as large as the largest type of operand in the equation, which is here a float.

75. B,E,F

Java keywords are words that are built into the Java language, and have special meanings, and therefore cannot be used as identifiers.

76. B

The sum of the variables “numA” and “numb” would be an int, and the variable to hold the sum is an int, so this works fine. The sum of the two numbers is 30, so the value of “numb” is printed.

77. C

The “>>” operator can only be used with integral types, and not with floating point types. So the statement “y = x >> 3;” will cause a compilation error since it is trying to use the “>>” operator on a float variable.

78. A

In the Java language these keywords must always be in lowercase letters to be recognized in the proper way. A programmer may use these words with some uppercase values, and the code will compile, but it is not recommended to do so.

79. A

In the Java language all types have a fixed range for their size, so there is no need to check the size of a type.

80. A, B, C

Identifiers are names that the programmer gives to variables, classes, etc. The rules in A, B and C apply for them.

81. A, B, C

These are all true statements.

82. D

Answer D starts with an 0x, which represents the number in hexadecimal form, and it contains letters that are valid for a hexadecimal representation of a number.

Answers A and B are not legal since a number with a ‘0’ before it becomes an octal number, and when using the octal base the digits 8 and 9 are not available.

Answer C is not legal since the letter G is not a valid letter for a hexadecimal number.

Answer E is not legal since the letter A and the number 0 at the end have no meaning in the sequence.

Answer F is not legal since it would need either an 0x or an 0X placed before it to make it into a hexadecimal number

83. A, B, C, D

Answer A is legal since it represents a legal float value.

Answer B is legal since it represents the number 22 cast to a long.

Answer C is legal since it represents a hexadecimal number cast to a long.

Answer D is legal since it represents a regular decimal number cast to a long.

When putting the letter F after a number the number is cast to a float, and when putting the letter L after a number the number is cast to a long. So, placing these letters at the end of a number is valid in A, B, C and D. In C the 0X in the front of the number is OK, since it refers to a hexadecimal number.

Answer E is incorrect since the letter A cannot be put at the front of a number.

Answer F is incorrect since it would need either an 0x or an 0X in the beginning to make it a hexadecimal number.

84. D

A compilation error will occur at line 7 when trying to assign the sum of “numA + numB” to “numB”. This is because “numB” is of type byte, and the sum will be of type int.

85. D

A compilation error will occur at line 8 since “sizeof” is not recognized in the Java language.

86. A

In the switch block “numA” has the value of 10, so when run the program goes into the first case statement, case 10. The variable “numA” is then increased by 1 to be 11. Since there are no break statements in all of the cases, the program goes into all of them automatically, regardless of the case value, and keeps adding 1 to “numA”. The result is then 14.

87. D

A compilation error will occur at line 6 when trying to put the sum of two numbers into a byte variable, when the sum is of type int. A cast to an int would be required to fix this code.

88. A

With the “?” operator, since the variable value of “num” is not greater than 10 the second value after the “?” is used, the 9. Then the value of “num” is decreased by 1 before printing it out, so the result is 8.

89. A

In the “while” loop the variable “i” increased by 20 each time. The first time into the loop both “sum” and “I” have the value of 0. The second

time in the loop, the value of “sum” is still 0, and the value of “i” is 20. The 20 is then added to “sum”, so “sum” now holds 20 as well. The value of “i” is then increased to 40 and the loop ends, leaving “sum” to have the value of 20.

90. A

At line 12 “numA” gets the value 70 assigned to it, and in line 14 “numC” is increased by 100, making it 130. Then in line 17 “numC” adds the value of “numA” to it, which is 130 + 70, which is 200.

91. A

The line “f1 = 3/2” results in an int with the value of 1, since when you divide an int by an int the result is an int. Since the variable that holds the result is a float, the number 1 is converted to a float and becomes 1.0.

The line “(float)3/2” turns the result into a float, so the result is 1.5.

Due to the parenthesis the line “(float)(3/2)” first gets the result of 3/2, which is 1, and then converts it to a float, which is 1.0.

So the result that is printed to the screen is 1.0+1.5+1.0 = 3.5.

92. E

The maximum positive number that a byte can hold is +127 and the maximum negative number that a byte can hold is –128. The numbers here are larger than this, so they are both considered to be int numbers, so both lines 6 and 7 would not compile since they are trying to place int values inside of byte variables.

93. D

A byte contains 8 bits and can hold positive and negative numbers within the range –128 to 127.

94. B

A byte type is always signed.