Section 4.2 Common Mathematical Functions

Section 4.2.1 Trigonometric Methdos

1. To obtain the sine of 35 degrees, use \_\_\_\_\_\_\_.

a. Math.sin(35)

b. Math.sin(Math.toRadians(35))

c. Math.sin(Math.toDegrees(35))

d. Math.sin(Math.toRadian(35))

e. Math.sin(Math.toDegree(35))

Key:b Note the trig methods use the radians for angles.

#

2. To obtain the arc sine of 0.5, use \_\_\_\_\_\_\_.

a. Math.asin(0.5)

b. Math.asin(Math.toDegrees(0.5))

c. Math.sin(Math.toRadians(0.5))

d. Math.sin(0.5)

Key:a Note the trig methods use the radians for angles.

#

3. Math.asin(0.5) returns \_\_\_\_\_\_\_.

a. 30

b. Math.toRadians(30)

c. Math.PI / 4

d. Math.PI / 2

Key:b Note that Math.asin returns an angle in radians.

#

4. Math.sin(Math.PI) returns \_\_\_\_\_\_\_.

a. 0.0

b. 1.0

c. 0.5

d. 0.4

Key:a Note that Math.PI is 180 degrees.

#

5. Math.cos(Math.PI) returns \_\_\_\_\_\_\_.

a. 0.0

b. 1.0

c. -1.0

d. 0.5

Key:c Note that Math.PI is 180 degrees.

#

Section 4.2.3 The Rounding Methods

5. What is Math.round(3.6)?

a. 3.0

b. 3

c. 4

d. 4.0

Key:c Note that round returns an int value

#

6. What is Math.rint(3.6)?

a. 3.0

b. 3

c. 4.0

d. 5.0

Key:c Note that rint returns a double value

#

7. What is Math.rint(3.5)?

a. 3.0

b. 3

c. 4

d. 4.0

e. 5.0

Key:d rint returns the nearest even integer as a double since 3.5 is equally close to 3.0 and 4.0.

#

8. What is Math.ceil(3.6)?

a. 3.0

b. 3

c. 4.0

d. 5.0

Key:c Note that ceil returns a double value

#

9. What is Math.floor(3.6)?

a. 3.0

b. 3

c. 4

d. 5.0

Key:a Note that floor returns a double value

#

Section 4.3 Character Data Type and Operations

Section 4.3.1 Unicode and ASCII Code

10. Which of the following is the correct expression of character 4?

a. 4

b. "4"

c. '\0004'

d. '4'

Key:d You have to write '4'.

#

11. A Java character is stored in \_\_\_\_\_\_\_\_\_\_.

a. one byte

b. two bytes

c. three bytes

d. four bytes

Key:b Java characters use Unicode encoding.

#

12. The Unicode of 'a' is 97. What is the Unicode for 'c'?

a. 96

b. 97

c. 98

d. 99

Key:d The Unicode for letters and numbers are allocated in a natural order. So b is after a and c is after b, and so on.

#

Section 4.3.2 Escape Sequences for Special Characters

13. Which of the following statement prints smith\exam1\test.txt?

a. System.out.println("smith\exam1\test.txt");

b. System.out.println("smith\\exam1\\test.txt");

c. System.out.println("smith\"exam1\"test.txt");

d. System.out.println("smith"\exam1"\test.txt");

Key:b To represent the \ character, use \\, because it is an escape character.

#

Section 4.3.3 Casting between char and Numeric Types

14. Suppose x is a char variable with a value 'b'. What is the output of the statement System.out.println(++x)?

a. a

b. b

c. c

d. d

Key:c The ++ and -- operators can be applied to a char variable. ++x is preincrement. x is 'b' before ++x. After ++x, x becomes c.

#

15. Suppose i is an int type variable. Which of the following statements display the character whose Unicode is stored in variable i?

a. System.out.println(i);

b. System.out.println((char)i);

c. System.out.println((int)i);

d. System.out.println(i + " ");

Key:b (char)i casts a number into a character.

#

16. Will System.out.println((char)4) display 4?

a. Yes

b. No

Key:b The character whose Unicode is \u0004 is to be displayed, not number 4.

#

17. What is the output of System.out.println('z' - 'a')?

a. 25

b. 26

c. a

d. z

Key:a The Unicode offset between z and a is 25.

#

18. An int variable can hold \_\_\_\_\_\_\_\_\_\_.

a. 'x'

b. 120

c. 120.0

d. "x"

e. "120"

Key:ab Choice A is also correct, because a character can be implicitly cast into an int variable. The Unicode value of character is assignment to the int variable. In this case, the code is 120 (see Appendix B).

#

19. Which of the following assignment statements is correct?

a. char c = 'd';

b. char c = 100;

c. char c = "d";

d. char c = "100";

Key:ab Choice B is also correct, because an int value can be implicitly cast into a char variable. The Unicode of the character is the int value. In this case, the character is d (see Appendix B).

#

20. '3' - '2' + 'm' / 'n' is \_\_\_\_\_\_.

a. 0

b. 1

c. 2

d. 3

Key:b When an operand is a character in an arithmetic expression, the character is casted to an int value.

#

Section 4.3.4 Comparing and Testing Characters

21. To check whether a char variable ch is an uppercase letter, you write \_\_\_\_\_\_\_\_\_\_\_.

a. (ch >= 'A' && ch >= 'Z')

b. (ch >= 'A' && ch <= 'Z')

c. (ch >= 'A' || ch <= 'Z')

d. ('A' <= ch <= 'Z')

Key:b A is wrong because ch >= 'Z'. C is wrong because of using ||. D is wrong because of incorrect syntax. The correct answer is B.

#

22. Which of the following is not a correct method in the Character class?

a. isLetterOrDigit(char)

b. isLetter(char)

c. isDigit()

d. toLowerCase(char)

e. toUpperCase()

Key:ce isDigit() should be isDigit(char) and toUpperCase() should be toUpperCase(char)

#

23. Suppose Character x = new Character('a'), \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ returns true.

a. x.equals(new Character('a'))

b. x.compareToIgnoreCase('A')

c. x.equalsIgnoreCase('A')

d. x.equals('a')

e. x.equals("a")

Key:ad (B) and (C) are wrong because no methods compareToIgnoreCase and equalsIgnoreCase are in the Character class. (E) is wrong because a character is not a string.

#

Section 4.4 The String Type

Section 4.4.2 Gettiing Characters from a String

24. Suppose s is a string with the value "java". What will be assigned to x if you execute the following code?

char x = s.charAt(4);

a. 'a'

b. 'v'

c. Nothing will be assigned to x, because the execution causes the runtime error StringIndexOutofBoundsException.

Key:c The string index starts from 0 and the last index is s.length() - 1. s.charAt(4) is out of bounds.

#

Section 4.4.3 Concatenating Strings

25. The expression "Java " + 1 + 2 + 3 evaluates to \_\_\_\_\_\_\_\_.

a. Java123

b. Java6

c. Java 123

d. java 123

e. Illegal expression

key:c The + operator is evaluated from left to right. When a string adds with a number, the number is converted into a string. The correct answer is C.

#

26. Note that the Unicode for character A is 65. The expression "A" + 1 evaluates to \_\_\_\_\_\_\_\_.

a. 66

b. B

c. A1

d. Illegal expression

key:c When a string adds with a number, the number is converted into a string. The correct answer is C.

#

27. Note that the Unicode for character A is 65. The expression 'A' + 1 evaluates to \_\_\_\_\_\_\_\_.

a. 66

b. B

c. A1

d. Illegal expression

key: a When a character adds with a number, the character is converted into a int. The correct answer is A.

#

Section 4.4.4 Converting Strings

28. Which of the following is the correct statement to return JAVA?

a. toUpperCase("Java")

b. "Java".toUpperCase("Java")

c. "Java".toUpperCase()

d. String.toUpperCase("Java")

Key:c The correct method is toUpperCase(). So C is correct.

#

Section 4.4.7 Comparing Strings

29. Suppose s1 and s2 are two strings. Which of the following statements or expressions is incorrect?

a. String s3 = s1 - s2;

b. boolean b = s1.compareTo(s2);

c. char c = s1[0];

d. char c = s1.charAt(s1.length());

Key:abcd A is wrong because the - operator cannot be used for strings. B is wrong because the compareTo method returns an int, not a boolean. C is wrong because the [] cannot be used for accessing string elements. D is wrong because of index out of bounds.

#

30. Suppose s1 and s2 are two strings. What is the result of the following code?

s1.equals(s2) == s2.equals(s1)

a. true

b. false

Key:a s1.equals(s2) and s2.equals(s1) are the same.

#

31. "abc".compareTo("aba") returns \_\_\_\_\_\_\_\_\_\_\_.

a. 1

b. 2

c. -1

d. -2

e. 0

Key:b The first two characters in the two strings are the same. The different between the last two characters is 2. The correct answer is B.

#

32. "AbA".compareToIgnoreCase("abC") returns \_\_\_\_\_\_\_\_\_\_\_.

a. 1

b. 2

c. -1

d. -2

e. 0

Key:d Ignoring case, you compare aba with abc. The first two characters in the two strings are the same. The different between the last two characters is -2. The correct answer is D.

#

33. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ returns true.

a. "peter".compareToIgnoreCase("Peter")

b. "peter".compareToIgnoreCase("peter")

c. "peter".equalsIgnoreCase("Peter")

d. "peter".equalsIgnoreCase("peter")

e. "peter".equals("peter")

Key:cde The compareToIgnoreCase return an int. So, A and B are wrong. Ignoring case, C, D, and E all return true.

#

Section 4.4.8 Obtaining Substrings

34. What is the return value of "SELECT".substring(0, 5)?

a. "SELECT"

b. "SELEC"

c. "SELE"

d. "ELECT"

Key:b Note that the substring is from index 0 to 4, which is 5 - 1. The correct answer is B.

#

35. What is the return value of "SELECT".substring(4, 4)?

a. an empty string

b. C

c. T

d. E

Key:a If beginIndex is endIndex, substring(beginIndex, endIndex) returns an empty string with length 0. It would be a runtime error, if beginIndex > endIndex.

#

Section 4.4.9 Finding a Character or a Substring in a String

36. To check if a string s contains the prefix "Java", you may write

a. if (s.startsWith("Java")) ...

b. if (s.indexOf("Java") == 0) ...

c. if (s.substring(0, 4).equals("Java")) ...

d. if (s.charAt(0) == 'J' && s.charAt(1) == 'a' && s.charAt(2) == 'v' && s.charAt(3) == 'a') ...

Key:abcd They are all correct.

#

37. To check if a string s contains the suffix "Java", you may write

a. if (s.endsWith("Java")) ...

b. if (s.lastIndexOf("Java") >= 0) ...

c. if (s.substring(s.length() - 4).equals("Java")) ...

d. if (s.substring(s.length() - 5).equals("Java")) ...

e. if (s.charAt(s.length() - 4) == 'J' && s.charAt(s.length() - 3) == 'a' && s.charAt(s.length() - 2) == 'v' && s.charAt(s.length() - 1) == 'a') ...

Key:ace s.lastIndexOf("Java") >= 0 does not indicate that Java is the suffix of the string.

#

Section 4.4.10 Conversions between Strings and Numbers

38. The \_\_\_\_\_\_\_\_\_\_ method parses a string s to an int value.

a. integer.parseInt(s);

b. Integer.parseInt(s);

c. integer.parseInteger(s);

d. Integer.parseInteger(s);

Key:b The parseInt method is defined in the Integer class. B is correct.

#

39. The \_\_\_\_\_\_\_\_\_\_ method parses a string s to a double value.

a. double.parseDouble(s);

b. Double.parsedouble(s);

c. double.parse(s);

d. Double.parseDouble(s);

Key:d The parseDouble method is defined in the Double class. D is correct.

#

Section 4.6 Formatting Console Output

40. Which of the following are valid specifiers for the printf statement?

a. %4c

b. %10b

c. %6d

d. %8.2d

e. %10.2e

Key:abce All correct. D is wrong because the specifier d is for decimal integer.

#

41. The statement System.out.printf("%3.1f", 1234.56) outputs \_\_\_\_\_\_\_\_\_\_\_.

a. 123.4

b. 123.5

c. 1234.5

d. 1234.56

e. 1234.6

Key:e .1 specifies one digit after the decimal point. The rest is rounded up. So 1234.56 is displayed 1234.6.

#

42. The statement System.out.printf("%3.1e", 1234.56) outputs \_\_\_\_\_\_\_\_\_\_\_.

a. 0.1e+04

b. 0.123456e+04

c. 0.123e+04

d. 1.2e+03

e. 1.23+03

Key:d %3.1e specifies a scientific notation with one digit after the decimal point. So, the correct answer is D.

#

43. The statement System.out.printf("%5d", 123456) outputs \_\_\_\_\_\_\_\_\_\_\_.

a. 12345

b. 23456

c. 123456

d. 12345.6

Key:c %5d specifies an integer with width 5. The width is automatically expanded if the number is larger than the specified width. So, the correct answer is C.

#

44. The statement System.out.printf("%10s", 123456) outputs \_\_\_\_\_\_\_\_\_\_\_. (Note: \* represents a space)

a. 123456\*\*\*\*

b. 23456\*\*\*\*\*

c. 12345\*\*\*\*\*

d. \*\*\*\*123456

Key:d %10s specifies to display a string with width 10. By default, it is right justified. So, the correct answer is D.

#

45. Analyze the following code:

int i = 3434; double d = 3434;

System.out.printf("%5.1f %5.1f", i, d);

a. The code compiles and runs fine to display 3434.0 3434.0.

b. The code compiles and runs fine to display 3434 3434.0.

c. i is an integer, but the format specifier %5.1f specifies a format for double value. The code has an error.

Key:c i is an integer, but the format specifier %5.1f specifies a format for double value. Type does not match. So, the correct answer is C.