## 1. Какие из следующих выражения являются допустимыми идентификаторами Java? (Выберите все подходящие варианты)

**A.** A$B

**B.** \_helloWorld

**C.** true

**D.** java.lang

**E.** Public

**F.** 1980\_s

## 2. Каков вывод следующей программы?

1: public class WaterBottle {

2: private String brand;

3: private boolean empty;

4: public static void main(String[] args) {

5: WaterBottle wb = new WaterBottle();

6: System.out.print("Empty = " + wb.empty);

7: System.out.print(", Brand = " + wb.brand);

8: }

9: }

**A.** Строка 6 генерирует ошибку на этапе компиляции.

**B.** Строка 7 генерирует ошибку на этапе компиляции.

**C.** Нет вывода.

**D.** Empty = false, Brand = null

**E.** Empty = false, Brand =

**F.** Empty = null, Brand = null

## 3. Что из перечисленного верно? (Выберите все подходящие варианты)

4: short numPets = 5;

5: int numGrains = 5.6;

6: String name = "Scruffy";

7: numPets.length();

8: numGrains.length();

9: name.length();

**A**. Строка 4 генерирует ошибку на этапе компиляции.

**B**. Строка 5 генерирует ошибку на этапе компиляции.

**C**. Строка 6 генерирует ошибку на этапе компиляции.

**D**. Строка 7 генерирует ошибку на этапе компиляции.

**E**. Строка 8 генерирует ошибку на этапе компиляции.

**F**. Строка 9 генерирует ошибку на этапе компиляции r.

**G**. Код успешно скомпилируется.

## 4. Какое из нижеследующих утвержденний истинно относительно класса Snake? (Выберите все верные варианты)

1: public class Snake {

2:

3: public void shed(boolean time) {

4:

5: if (time) {

6:

7: }

8: System.out.println(result);

9:

10: }

11: }

**A**. Если втсавить строку String result = "done"; вместо строки 2, то код скомпилируется.

**B**. Если вставить строку String result = "done"; вместо строки 4, то код скомпилируется.

**C.** Если вставить строку String result = "done"; вместо строки 6, то код скомпилируется.

**D**. Если вставить строку String result = "done"; вместро строки 9, то код скомпилируется.

**E**. Ни одно из вышеперечисленных изменений не позволит скомпилировать код.

## 5. Для заданных классов, какое выражение можно подтсавить вместо строки INSERT IMPORTS HERE для того, что бы этот код мог быть скомпилирован? (Выберите все возможные варианты)

package aquarium;

public class Tank { }

package aquarium.jellies;

public class Jelly { }

package visitor;

INSERT IMPORTS HERE

public class AquariumVisitor {

public void admire(Jelly jelly) { } }

**A**. import aquarium.\*;

**B**. import aquarium.\*.Jelly;

**C**. import aquarium.jellies.Jelly;

**D**. import aquarium.jellies.\*;

**E**. import aquarium.jellies.Jelly.\*;

**F**. Ни одно из вышеперечисленных изменений не позволит скомпилировать код.

## 6. С учетом следующих классов, какое максимальное количество импортов можно безопасно удалить и при этом код все еще будет компилироваться?

package aquarium; public class Water { }

package aquarium;

import java.lang.\*;

import java.lang.System;

import aquarium.Water;

import aquarium.\*;

public class Tank {

public void print(Water water) {

System.out.println(water);

}

}

**A**. 0

**B**. 1

**C**. 2

**D**. 3

**E**. 4

**F**. Не компилируется.

## 7. С учетом представленных классов, какие из следующих фрагментов можно вставить вместо INSERT IMPORTS HERE и скомпилировать код? (Выберите все возможные варианты)

package aquarium;

public class Water {

boolean salty = false;

}

package aquarium.jellies;

public class Water {

boolean salty = true;

}

package employee;

INSERT IMPORTS HERE

public class WaterFiller {

Water water;

}

**A**. import aquarium.\*;

**B**. import aquarium.Water;

import aquarium.jellies.\*;

**C**. import aquarium.\*;

import aquarium.jellies.Water;

**D**. import aquarium.\*;

import aquarium.jellies.\*;

**E**. import aquarium.Water;

import aquarium.jellies.Water;

**F**. Ничего из вышеприведенного не позволит скомпилировать.

## 8. Что из перечисленного верно? (Выберите все возможные варианты)

**A.** Переменная экземпляра класса типа double по умолчанию null.

**B.** Переменная экземпляра класса типа int по умолчаниюn null.

**C.** Переменная экземпляра класса типа String по умолчаниюn null.

**D.** Переменная экземпляра класса типа double по умолчанию 0.0.

**E.** Переменная экземпляра класса типа int по умолчанию 0.0.

**F.** Переменная экземпляра класса типа String по умолчанию 0.0.

**G.** Ничего из вышеперечисленного не верно.

## 9. Что из перечисленного верно? (Выберите все возможные варианты)

**A**. Локальная переменная типа boolean по умолчанию null.

**B**. Локальная переменная типа float по умолчанию 0.

**C**. Локальная переменная типа Object по умолчанию null.

**D**. Локальная переменная типа boolean по умолчанию false.

**E**. Локальная переменная типа boolean по умолчанию true.

**F**. Локальная переменная типа float по умолчанию 0.0.

**G**. Ничего из вышеперечисленного не верно.

## 10. Что из перечисленного верно? (Выберите все возможные варианты)

**A**. Переменная экземпляра класса типа boolean по умолчанию false.

**B**. Переменная экземпляра класса типа boolean по умолчанию true.

**C**. Переменная экземпляра класса типа boolean по умолчанию null.

**D**. Переменная экземпляра класса типа int по умолчанию 0.

**E**. Переменная экземпляра класса типа int по умолчанию 0.0.

**F**. Переменная экземпляра класса типа int по умолчанию null.

**G**. Ничего из вышеперечисленного не верно.

## 11. Дан класс в файле /my/directory/named/A/Bird.java:

INSERT CODE HERE

public class Bird { }

## Чем может быть заменена строка INSERT CODE HERE если компиляция выполняется в директории /my/directory? (Выберите все возможные варианты)

**A**. package my.directory.named.a;

**B**. package my.directory.named.A;

**C**. package named.a;

**D**. package named.A;

**E.** package a;

**F.** package A;

**G**. Не комплилируется.

## 12. Какие из следующих строк пройдут компиляцию? (Выберите все возможные варианты)

**A**. int i1 = 1\_234;

**B**. double d1 = 1\_234\_.0;

**C**. double d2 = 1\_234.\_0;

**D**. double d3 = 1\_234.0\_;

**E**. double d4 = 1\_234.0;

**F**. Ничего из выше перечисленного.

## 13. Дан класс Price, что может быть подставлено на место строки INSERT CODE HERE для успешной компиляции класса? (Выберите все возможные варианты)

public class Price {

public void admission() {

INSERT CODE HERE

System.out.println(amount);

}

}

**A.** int amount = 9L;

**B**. int amount = 0b101;

**C**. int amount = 0xE;

**D**. double amount = 0xE;

**E**. double amount = 1\_2\_.0\_0;

**F**. int amount = 1\_2\_;

**G**. None of the above.

## 14. Какие из следующих утверждений правильные? (Выберите все возможные варианты)

public class Bunny {

public static void main(String[] args) {

Bunny bun = new Bunny();

}

}

**A**. Bunny – это класс.

**B**. bun – это класс.

**C**. main – это класс.

**D**. Bunny – это ссылка на объект.

**E**. bun – это ссылка на объект.

**F**. main – это ссылка на объект.

**G**. Ничего из выше перечисленного.

## 15. В какой последовательности необходимо разместить строки кода для успешной компиляции класса? (Выберите все возможные варианты)

A: class Rabbit {}

B: import java.util.\*;

C: package animals;

**A.** A, B, C

**B.** B, C, A

**C.** C, B, A

**D.** B, A

**E.** C, A

**F.** A, C

**G.** A, B

## 16. Есть класс Rabbit. Какие из следующих утверждений верны? (Выберите все возможные варианты)

1: public class Rabbit {

2: public static void main(String[] args) {

3: Rabbit one = new Rabbit();

4: Rabbit two = new Rabbit();

5: Rabbit three = one;

6: one = null;

7: Rabbit four = one;

8: three = null;

9: two = null;

10: two = new Rabbit();

11: System.gc();

12: } }

A. Сборщик мусора сразу освободит память на строке 6 выделенную для объекта Rabbit созданного на строке 3.

B. Сборщик мусора сразу освободит память на строке 8 выделенную для объекта Rabbit созданного на строке 3.

C. Сборщик мусора сразу освободит память на строке 12 выделенную для объекта Rabbit созданного на строке 3.

D. Сборщик мусора сразу освободит память на строке 9 выделенную для объекта Rabbit созданного на строке 4.

E. Сборщик мусора сразу освободит память на строке 11 выделенную для объекта Rabbit созданного на строке 4.

F. Сборщик мусора сразу освободит память на строке 12 выделенную для объекта Rabbit созданного на строке 4.

## 17. Какие из следующих утверждений верны в отношении нижеследующего кода? (Выберите все возможные варианты)

public class Bear {

protected void finalize() {

System.out.println("Roar!");

}

public static void main(String[] args) {

Bear bear = new Bear();

bear = null;

System.gc();

}

}

**A**. Метод finalize() точно будет вызван.

**B**. Метод finalize() может выть вызван, а может быть и не вызван.

**C**. Метод finalize() точно не будет вызван.

**D**. Сборщик мусора точно будет запущен.

**E**. Сборщик мусора может быть запущен, а может быть и не запущен.

**F**. Сборщик мусора точно не запустится.

**G**. Код не скомпилируется.

## 18. Каков вывод следующего кода?

1: public class Salmon {

2: int count;

3: public void Salmon() {

4: count = 4;

5: }

6: public static void main(String[] args) {

7: Salmon s = new Salmon();

8: System.out.println(s.count);

9: } }

**A**. 0

**B**. 4

**C**. Ошибка компиляции на строке 3.

**D**. Ошибка компиляции на строке 4.

**E**. Ошибка компиляции на строке 7.

**F**. Ошибка компиляции на строке 8.

## 19. Какие из следующих утвержедний верны? (Выберите все возможные варианты)

A. В Java есть возможность перерузки операторов.

B. Код Java скомпилированный в Windows может быть запущен в Linux.

C. Java имеет указатели для прямого обращения к памяти.

D. Java – это процедурный язык программировани.

E. Java – это объектно ориентированный язык программирования.

F. Java – это функциональный язык программирования.

## 20. Какие из следующих утвержедний верны? (Выберите все возможные варианты)

A. javac компилирует файл .class в файл .java.

B. javac компилирует файл .java в файл .bytecode.

C. javac компилирует файл .java в файл .class.

D. Java принимает имя класса как параметр.

E. Java принимает имя файла .bytecode как параметр.

F. Java принимает имя класса .class как параметр.

## 21. Какие из следующих операторов Java можно использовать с булевыми переменными? (Выберите все возможные варианты)

**A**. ==

**B**. +

**C**. --

**D**. !

**E**. %

**F**. <=

## 22. Какой тип данных (или типы) необходимо подставить на место прочерка для успешной компиляции следующего фрагмента кода? (Выберите все возможные варианты)

byte x = 5;

byte y = 10;

\_\_\_\_\_ z = x + y;

**A**. int

**B**. long

**C**. boolean

**D**. double

**E**. short

**F**. byte

## 23. Каков результат работы следующего приложения?

1: public class CompareValues {

2: public static void main(String[] args) {

3: int x = 0;

4: while(x++ < 10) {}

5: String message = x > 10 ? "Greater than" : false;

6: System.out.println(message+","+x);

7: }

8: }

**A**. Greater than,10

**B**. false,10

**C**. Greater than,11

**D**. false,11

**E**. Этот код не скомпилируется из-за ошибки на строке 4.

**F**. Этот код не скомпилируется из-за ошибки на строке 5.

## 24. Какие правки необходимо внести, чтобы код успешно скомпилировался? (Выберите все возможные варианты)

3: long x = 10;

4: int y = 2 \* x;

**A**. Ничего не нужно изменять; код скомпилируется.

**B**. Привести значение переменной x на строке 4 к int.

**C**. Изменить тип данных переменной x на строке 3 на short.

**D**. Привести 2 \* x на строке 4 к int.

**E**. Изменить тип данных переменной y на строке 4 к short.

**F.** Изменить тип данных переменной y на строке 4 к long.

## 25. Каков результат следующего фрагмента кода?

3: java.util.List<Integer> list = new java.util.ArrayList<Integer>();

4: list.add(10);

5: list.add(14);

6: for(int x : list) {

7: System.out.print(x + ", ");

8: break;

9: }

**A**. 10, 14,

**B**. 10, 14

**C**. 10,

**D**. Этот код не скомпилируется из-за ошибки на строке 7.

**E**. Этот код не скомпилируется из-за ошибки на строке 8.

**F**. Этот код содержит бесконечный цикл и не завершается.

## 26. Каков результат следующего фрагмента кода?

3: int x = 4;

4: long y = x \* 4 - x++;

5: if(y<10) System.out.println("Too Low");

6: else System.out.println("Just right");

7: else System.out.println("Too High");

**A**. Too Low

**B**. Just Right

**C**. Too High

**D**. Скомпилируется, но выбросит исключение NullPointerException.

**E**. Этот код не скомпилируется из-за ошибки на строке 6.

**F**. Этот код не скомпилируется из-за ошибки на строке 7.

## 27. Каков результат следующего фрагмента кода?

1: public class TernaryTester {

2: public static void main(String[] args) {

3: int x = 5;

4: System.out.println(x > 2 ? x < 4 ? 10 : 8 : 7);

5: }

6:}

**A**. 5

**B**. 4

**C**. 10

**D**. 8

**E**. 7

**F**. Этот код не скомпилируется из-за ошибки на строке 4.

## 28. Каков результат следующего фрагмента кода?

3: boolean x = true, z = true;

4: int y = 20;

5: x = (y != 10) ^ (z=false);

6: System.out.println(x+", "+y+", "+z);

**A**. true, 10, true

**B**. true, 20, false

**C**. false, 20, true

**D**. false, 20, false

**E.** false, 20, true

**F**. Этот код не скомпилируется из-за ошибки на строке 5.

## 29. Сколько раз этот код напечатает "Hello World"?

3: for(int i=0; i<10 ; ) {

4: i = i++;

5: System.out.println("Hello World");

6: }

**A**. 9

**B**. 10

**C**. 11

**D**. Этот код не скомпилируется из-за ошибки на строке 3.

**E**. Этот код не скомпилируется из-за ошибки на строке 5.

**F**. Этот код содержит бесконечный цикл и не завершается.

## 30. Каков результат следующего фрагмента кода?

3: byte a = 40, b = 50;

4: byte sum = (byte) a + b;

5: System.out.println(sum);

A. 40

B. 50

C. 90

D. Этот код не скомпилируется из-за ошибки на строке 4.

E. Неопределенное значение.

## 31. Каков результат следующего фрагмента кода?

1: public class ArithmeticSample {

2: public static void main(String[] args) {

3: int x = 5 \* 4 % 3;

4: System.out.println(x);

5: }

6: }

**A**. 2

**B**. 3

**C**. 5

**D**. 6

**E**. Этот код не скомпилируется из-за ошибки на строке 3.

## 32. Каков результат следующего фрагмента кода?

3: int x = 0;

4: String s = null;

5: if(x == s) System.out.println("Success");

6: else System.out.println("Failure");

**A**. Success

**B**. Failure

**C**. Этот код не скомпилируется из-за ошибки на строке 4.

**D**. Этот код не скомпилируется из-за ошибки на строке 5.

## 33. Каков результат следующего фрагмента кода?

3: int x1 = 50, x2 = 75;

4: boolean b = x1 >= x2;

5: if(b = true) System.out.println("Success");

6: else System.out.println("Failure");

A. Success

B. Failure

C. Этот код не скомпилируется из-за ошибки на строке 4.

D. Этот код не скомпилируется из-за ошибки на строке 5.

## 34. Каков результат следующего фрагмента кода?

3: int c = 7;

4: int result = 4;

5: result += ++c;

6: System.out.println(result);

**A**. 8

**B**. 11

**C**. 12

**D**. 15

**E**. 16

**F**. Этот код не скомпилируется из-за ошибки на строке 5.

## 35. Каков результат следующего фрагмента кода?

3: int x = 1, y = 15;

4: while x < 10

5: y––;

6: x++;

7: System.out.println(x+", "+y);

**A**. 10, 5

**B**. 10, 6

**C.** 11, 5

**D**. Этот код не скомпилируется из-за ошибки на строке 3.

**E**. Этот код не скомпилируется из-за ошибки на строке 4.

**F**. Этот код содержит бесконечный цикл и не завершается.

## 36. Каков результат следующего фрагмента кода?

3: do {

4: int y = 1;

5: System.out.print(y++ + " ");

6: } while(y <= 10);

**A**. 1 2 3 4 5 6 7 8 9

**B**. 1 2 3 4 5 6 7 8 9 10

**C**. 1 2 3 4 5 6 7 8 9 10 11

**D**. Этот код не скомпилируется из-за ошибки на строке 6.

**E**. Этот код содержит бесконечный цикл и не завершается.

## 37. Каков результат следующего фрагмента кода?

3: boolean keepGoing = true;

4: int result = 15, i = 10;

5: do {

6: i--;

7: if(i==8) keepGoing = false;

8: result -= 2;

9: } while(keepGoing);

10: System.out.println(result);

**A**. 7

**B**. 9

**C**. 10

**D**. 11

**E**. 15

**F**. Этот код не скомпилируется из-за ошибки на строке 8.

## 38. Каков результат следующего фрагмента кода?

3: int count = 0;

4: ROW\_LOOP: for(int row = 1; row <=3; row++)

5: for(int col = 1; col <=2 ; col++) {

6: if(row \* col % 2 == 0) continue ROW\_LOOP;

7: count++;

8: }

9: System.out.println(count);

**A**. 1

**B**. 2

**C**. 3

**D**. 4

**E**. 6

**F**. Этот код не скомпилируется из-за ошибки на строке 6.

## 39. Каков результат следующего фрагмента кода?

3: int m = 9, n = 1, x = 0;

4: while(m > n) {

5: m--;

6: n += 2;

7: x += m + n;

8: }

9: System.out.println(x);

**A**. 11

**B**. 13

**C**. 23

**D**. 36

**E**. 50

**F**. Этот код не скомпилируется из-за ошибки на строке 7.

## 40. Каков результат следующего фрагмента кода?

3: final char a = 'A', d = 'D';

4: char grade = 'B';

5: switch(grade) {

6: case a:

7: case 'B': System.out.print("great");

8: case 'C': System.out.print("good"); break;

9: case d:

10: case 'F': System.out.print("not good");

11: }

**A**. great

**B**. greatgood

**C**. Этот код не скомпилируется из-за ошибки на строке 3.

**D**. Этот код не скомпилируется из-за ошибки на строке 6.

**E**. Этот код не скомпилируется из-за ошибки на строке 9.

## 41. Что нужно поставить на место прочерка для компиляции этого фрагмента кода? (Укажите все возможные варианты)

public class Ant {

\_\_\_\_\_ void method() { }

}

**A**. default

**B**. final

**C**. private

**D**. Public

**E**. String

**F**. zzz:

## 42. Что скомпилируется? (Укажите все возможные варианты)

A. final static void method4() { }

B. public final int void method() { }

C. private void int method() { }

D. static final void method3() { }

E. void final method() {}

F. void public method() { }

## 43. Какие методы скомпилируются? (Укажите все возможные варианты)

A. public void methodA() { return;}

B. public void methodB() { return null;}

C. public void methodD() {}

D.public int methodD() { return 9;}

E. public int methodE() { return 9.0;}

F. public int methodF() { return;}

G. public int methodG() { return null;}

## 44. Что скомпилируется? (Укажите все возможные варианты)

A. public void moreA(int... nums) {}

B. public void moreB(String values, int... nums) {}

C. public void moreC(int... nums, String values) {}

D. public void moreD(String... values, int... nums) {}

E. public void moreE(String[] values, ...int nums) {}

F. public void moreF(String... values, int[] nums) {}

G. public void moreG(String[] values, int[] nums) {}

## 45. Дан метод howMany(), какой вызов метода вернет значение 2? (Укажите все возможные варианты)

public int howMany(boolean b, boolean... b2) {

return b2.length;

}

A. howMany();

B. howMany(true);

C. howMany(true, true);

D. howMany(true, true, true);

E. howMany(true, {true});

F. howMany(true, {true, true});

G. howMany(true, new boolean[2]);

## 46. Дано определение классов my.school.ClassRoom и my.city.School, какая строка в методе main() приведет к ошибке компиляции? (Укажите все возможные варианты)

1: package my.school;

2: public class Classroom {

3: private int roomNumber;

4: protected String teacherName;

5: static int globalKey = 54321;

6: public int floor = 3;

7: Classroom(int r, String t) {

8: roomNumber = r;

9: teacherName = t;

10: }

10: }

1: package my.city;

2: import my.school.\*;

3: public class School {

4: public static void main(String[] args) {

5: System.out.println(Classroom.globalKey);

6: Classroom room = new Classroom(101, ""Mrs. Anderson");

7: System.out.println(room.roomNumber);

8: System.out.println(room.floor);

9: System.out.println(room.teacherName);

10: }

11:}

**A**. Никакая, все успешно скомпилируется.

**B**. Строка 5

**C**. Строка 6

**D**. Строка 7

**E**. Строка 8

**F**. Строка 9

## 47. Что из перечисленного верно? (Укажите все возможные варианты)

A. Инкапсуляция достигается за счет применения package private переменных-членов класса.

B. Инкапсуляция достигается за счет применения private переменных-членов класса.

C. Инкапсуляция допускает использование сеттеров.

D. Неизменность класса достигается за счет применения package private переменных-членов класса.

E. Неизменность класса достигается за счет применения private переменных-членов класса.

F. Неизменные классы допускают использование сеттеров.

## 48. Каков результат следующего фрагмента кода?

1: package rope;

2: public class Rope {

3: public static int LENGTH = 5;

4: static {

5: LENGTH = 10;

6: }

7: public static void swing() {

8: System.out.print("swing ");

9: }

10: }

1: import rope.\*;

2: import static rope.Rope.\*;

3: public class Chimp {

4: public static void main(String[] args) {

5: Rope.swing();

6: new Rope().swing();

7: System.out.println(LENGTH);

8: }

9: }

A. swing swing 5

B. swing swing 10

C. Ошибка компиляции на строке 2 в классе Chimp.

D. Ошибка компиляции на строке 5 в классе Chimp.

E. Ошибка компиляции на строке 6 в классе Chimp.

F. Ошибка компиляции на строке 7 в классе Chimp.

## 49. Какие утверждения верны относительно этого кода? (Укажите все возможные варианты)

1: public class Rope {

2: public static void swing() {

3: System.out.print("swing ");

4: }

5: public void climb() {

6: System.out.println("climb ");

7: }

8: public static void play() {

9: swing();

10: climb();

11: }

12: public static void main(String[] args) {

13: Rope rope = new Rope();

14: rope.play();

15: Rope rope2 = null;

16: rope2.play();

17: }

18: }

**A**. Код успешно скомпилируется.

**B**. Во время компиляции возникнет одна ошибка.

**C**. Во время компиляции возникнет две ошибки.

**D.** Если строки с ошибкой компиляции удалить, то вывод будет climb climb.

**E**. Если строки с ошибкой компиляции удалить, то вывод будет swing swing.

**F**. Если строки с ошибкой компиляции удалить, то вывод будет возникнет исключение NullPointerException.

## 50. Каков результат следующего фрагмента кода?

import rope.\*;

import static rope.Rope.\*;

public class RopeSwing {

private static Rope rope1 = new Rope();

private static Rope rope2 = new Rope();

{

System.out.println(rope1.length);

}

public static void main(String[] args) {

rope1.length = 2;

rope2.length = 8;

System.out.println(rope1.length);

}

}

package rope;

public class Rope {

public static int length = 0;

}

**A**. 02

**B**. 08

**C**. 2

**D**. 8

**E**. Код не скомпилируется.

**F**. Выбросит исключение.

## 51. Сколько ошибок компиляции в следующем коде?

1: public class RopeSwing {

2: private static final String leftRope;

3: private static final String rightRope;

4: private static final String bench;

5: private static final String name = "name";

6: static {

7: leftRope = "left";

8: rightRope = "right";

9: }

10: static {

11: name = "name";

12: rightRope = "right";

13: }

14: public static void main(String[] args) {

15: bench = "bench";

16: }

17: }

A. 0

B. 1

C. 2

D. 3

E. 4

F. 5

## 52. Что нужно подставить на место строки 2 для компиляции следующего кода? (Укажите все возможные варианты)

1: import java.util.\*;

2: // INSERT CODE HERE

3: public class Imports {

4: public void method(ArrayList<String> list) {

5: sort(list);

6: }

7: }

**A**. import static java.util.Collections;

**B**. import static java.util.Collections.\*;

**C**. import static java.util.Collections.sort(ArrayList<String>);

**D**. static import java.util.Collections;

**E**. static import java.util.Collections.\*;

**F**. static import java.util.Collections.sort(ArrayList<String>);

## 53. Каков результат следующего фрагмента кода?

1: public class Test {

2: public void print(byte x) {

3: System.out.print("byte");

4: }

5: public void print(int x) {

6: System.out.print("int");

7: }

8: public void print(float x) {

9: System.out.print("float");

10: }

11: public void print(Object x) {

12: System.out.print("Object");

13: }

14: public static void main(String[] args) {

15: Test t = new Test();

16: short s = 123;

17: t.print(s);

18: t.print(true);

19: t.print(6.789);

20: }

21: }

**A**. bytefloatObject

**B**. intfloatObject

**C**. byteObjectfloat

**D**. intObjectfloat

**E**. intObjectObject

**F**. byteObjectObject

## 54. Каков результат следующего фрагмента кода?

1: public class Squares {

2: public static long square(int x) {

3: long y = x \* (long) x;

4: x = -1;

5: return y;

6: }

7: public static void main(String[] args) {

8: int value = 9;

9: long result = square(value);

10: System.out.println(value);

11: }

12:}

A. -1

B. 9

C. 81

D. Ошибка компиляции на строке 9.

E. Ошибки компиляции на разных строках.

## 55. Каков результат следующего фрагмента кода? (Укажите все возможные варианты)

public class StringBuilders {

public static StringBuilder work(StringBuilder a, StringBuilder b){

a = new StringBuilder("a");

b.append("b");

return a;

}

public static void main(String[] args) {

StringBuilder s1 = new StringBuilder("s1");

StringBuilder s2 = new StringBuilder("s2");

StringBuilder s3 = work(s1, s2);

System.out.println("s1 = " + s1);

System.out.println("s2 = " + s2);

System.out.println("s3 = " + s3);

}

}

A. s1 = a

B. s1 = s1

C. s2 = s2

D. s2 = s2b

E. s3 = a

F. s3 = null

G. Код не скомпилируется.

## 56. Какие из следующих утверждение верны? (Выберите два варианта)

A. this() может быть вызван в любом месте конструктора.

B. this() может быть вызван в любом методе экземпляра класса.

C. this.variableName может быть вызван в любом методе экземпляра класса.

D. this.variableName может быть вызван в любом статическом методе класса.

E. Необходимо добавить конструктор по умолчанию если компилятор не добавляет ни одного.

F. Можно вызвать конструктор по умолчанию добавленный компилятором используя this().

G. Можно получить прямой доступ к закрытому (private) конструктору в методе main().

## 57. Укажите классы, которые компилируются и используют конструкторы по умолчанию? (Укажите все возможные варианты)

**A**. public class Bird { }

**B**. public class Bird { public bird() {} }

**C**. public class Bird { public bird(String name) {} }

**D**. public class Bird { public Bird() {} }

**E**. public class Bird { Bird(String name) {} }

**F**. public class Bird { private Bird(int age) {} }

**G**. public class Bird { void Bird() { }

## 58. Чем можно заменить строки комментария, что бы была напечатана цифра 2?

public class BirdSeed {

private int numberBags;

boolean call;

public BirdSeed() {

// LINE 1

call = false;

// LINE 2

}

public BirdSeed(int numberBags) {

this.numberBags = numberBags;

}

public static void main(String[] args) {

BirdSeed seed = new BirdSeed();

System.out.println(seed.numberBags);

}

}

**A**. Заменить строку 1 на BirdSeed(2);

**B**. Заменить строку 2 на BirdSeed(2);

**C**. Заменить строку 1 на new BirdSeed(2);

**D**. Заменить строку 2 на new BirdSeed(2);

**E.** Заменить строку 1 на this(2);

**F**. Заменить строку 2 на this(2);