Учебен център SoftAcad

website: www.softacad.bg
email: info@softacad.bg

Програмиране с Java - Тест

Решение

Всяка задача дава максимум 3 точки. Максимален брой точки – 87.

1. Какво ще изведе програмата?

```
public static void main(String[] args) {
	int sum = 0;
	for (int i = 0; i < 10; i=i+2) {
		sum += i;
	}
	System.out.println("The sum is: " + sum);
}
```

```
The sum is: 20
```

2. Какво ще изведе програмата?

```
public static void main(String[] args) {
	for (int i = 10; i > -500000; i--) {
		if(i % 2 == 0) {
			continue;
		}
		System.out.print(i + " ");
		if(i < 0)
			break;
	}
}
```

```
9 7 5 3 1 -1
```

3. Оправете програмата за да прочита правилно масив от конзолата:

```
public static void main(String[] args) {
	Scanner sc = new Scanner(System.in);
	System.out.println("Въведете размер на масива");
	int size = sc.nextInt();

	double array[] = new double[size];

	for (int i = 0; i < array.length; i++) {
		System.out.println("Въведете " + ( i + 1 ) + "-тия елемент:");
		array[i] = sc.nextDouble();
	}
}
```

4. Ще се компилира ли програмата? Ако да, какво ще се изведе на конзолата?

```
public class A {
	public static void main(String[] args) {
		String s = new String("Test String");
		String s1 = new String("Test ");
		s1 += "String";

		System.out.println(s);
		System.out.println(s1);

		if(s == s1) {
			System.out.println("The Strings are equals");
		}
	}
}
```

```
Test String
Test String
```

5. Какъв ще е изходът на програмата?

```
public static void main(String[] args) {
	String s = new String("Some String");
	String s1 = s;
	s1 = s1 + " some text";
	s.toUpperCase();

	System.out.println(s);
	System.out.println(s1);
}
```

Метода по никакъв начин не променя s. Той просто връща стойността на s с големи букви, която не се присвоява на нищо.

String-овете са immutable, т.е. не променят стойността си

```
Some String
Some String some text
```

6. Какво ще изведе програмата:

```
public class A {
	public static int x = 0;
	public int y;

	public A(int x, int y){
		this.x = x;
		this.y = y;
	}


	public static void main(String[] args) {
		A a1 = new A(2,3);
		A a2 = new A(7, 9);

		System.out.println(a2.y++);
		a1.x += a2.y;
		System.out.println(a1.x);
		a2.y = --a1.y;
		System.out.println(a2.y);
	}
}
```

Първо се принтира стойността на a2.y и после a2.y се увеличава с 1.

Първо стойността на a1.y се намалява с 1 и чак тогава се присвоява на a2.y

```
9
17
2
```

7. Даден е класът Human в пакета human:

```
package human;
public class Human {
	public String name;
	protected boolean sex;
	int age;
	private String EGN;
}
```

Подчертайте местата където възникват грешки при компилиране:

```
package human;
public class Child extends Human{
	Child (){
		super();
		age = 10;
	}

	Child (String EGN){
		this.EGN = EGN;
	}

}
```

private поле

```
package employee;
import human.Human;
public class Employee extends Human{
	double salary;

	void showInfo(){
		System.out.println(name);
		System.out.println(age);
		System.out.println(sex);
		System.out.println(EGN);
	}
}
```

поле с default достъп не може да се достъпва извън рамките на същия пакет

private поле

8. Дадена е следната йерархия от класове:

```
package car;
public class Car {
	public String model;

	public Car(String model) {
		this.model = model;
	}
}

package car;
public class BMW extends Car {
	public double maxSpeed;

	public BMW(double maxSpeed) {
		this.maxSpeed = maxSpeed;
	}
}
```

Няма извикване на конструктор на базовия клас. Трябва или да се извика изрично конструктора `Car(String model)` или в Car да се дефинира конструктор без параметри.

Какво ще се изведе след изпълнението на следния main метод:

```
package car;
```

```
public class CarDemo {

	public static void main(String[] args) {
		Car car = new Car("Mercedes");
		System.out.println("Model: " + car.model);

		BMW bmw = new BMW(290.0);
		System.out.println("Model: " + bmw.model);
		System.out.println("Max speed: " + bmw.maxSpeed);
	}
}
```

А Ще се изведе на конзолата:
```
Model: Mercedes
Max Speed: 290.0
```

Б Ще се изведе на конзолата:
```
Model: Mercedes
Model: Mercedes
Max Speed: 290.0
```

В Ще се хвърли exception на следния ред:
```
System.out.println("Model: " + bmw.model);
```

Г Ще се изведе на конзолата:
```
Model: Mercedes
Model: null
Max Speed: 290.0
```

Д Програмата няма да се компилира (ако изберете този отговор, напишете защо)

9. Даден е интерфейса:

```
interface Base {
  boolean m1 ();
  byte m2(short s);
}
```

Interface не може да имплементира, а само да наследява друг interface

Кой код ще се компилира?

А
```
interface Base2 implements Base { }
```

Клас не може да наследява, а само да имплементира interface

Б
```
abstract class Class2 extends Base {
	public boolean m1() { return true; }
}
```

В
```
abstract class Class2 implements Base { }
```

Г
```
abstract class Class2 implements Base {
	public boolean m1() { return (true); }
}
```

Двата метода трябва да бъдат public защото в интерфейса всички методи са public, а в подклас не можем да намаляваме нивото на видимост.

Д
```
class Class2 implements Base {
	boolean m1( ) { return false; }
	byte m2(short s) { return 42; }
}
```

10. Кой код дефинира правилно абстрактен клас?

Липсва abstract

А
```
public abstract class Canine { public Bark speak(); }
```

Класът има абстрактен метод, значи трябва да бъде маркиран с abstract

Б
```
public
```

В
```
public class Canine { public abstract Bark speak(); }
```

Г
```
public class Canine abstract { public abstract Bark speak(); }
```

Ключовата дума abstract е на грешно място

11. Даден е кода:

```
class Clidders {
	public final void flipper() {
		System.out.println("Clidder");
	}
}

public class Clidlets extends Clidders {
	public void flipper() {
		System.out.println("Flip a Clidlet");
		super.flipper();
	}

	public static void main(String[] args) {
		new Clidlets().flipper();
	}
}
```

final метод не може да се override-ва

Какъв ще е резултата?

А Flip a Clidlet

Б Flip a Clidder

В Flip a Clidder

Flip a Clidlet

Г Flip a Clidlet

Flip a Clidder

Е Компилационна грешка

12. Даден е интерфейса:

```
public interface Frobnicate {
	public void twiddle(String s) ;
}
```

Кои класове се компилират без грешка. За тези при които има грешки, дайте кратко обяснение.

А
```
public abstract class Frob implements Frobnicate {
	public abstract void twiddle(String s) { }
}
```

Абстрактен метод няма тяло

Б
```
public abstract class Frob implements Frobnicate { }
```

В
```
public class Frob extends Frobnicate {
	public void twiddle(int i) { }
}
```

Г
```
public class Frob implements F
	public void twiddle
```

Клас не може да наследява интерфейс

имплементира метода twiddle(String s), нито е обявен като абстрактен.

```
}
```

Д

```
public class Frob implements Frobnicate {
	public void twiddle(String i) { }
	public void twiddle(int s) { }
}
```

13. Дадено:

```
class Top {
	public Top(String s) {
		System.out.print("B");
	}
}

public class Bottom2 extends Top {
	public Bottom2(String s) {
		System.out.print("D");
	}

	public static void main(String[] args) {
		new Bottom2("C");
		System.out.println(" ");
	}
}
```

Няма извикване на конструктор на базовия клас.
Трябва или изрично да се извика конструктора Top(String s) чрез ключовата дума super или да се дефинира конструктор без параметри в класа Top.

Какъв ще е изведения резултата?

А BD

Б DB

В BDC

Г DBC

Д Компилационна грешка

Тук НЯМА компилационна грешка, тъй като метода на класа Human не е видим тук и този код дефинира нов метод с име enjoy.
Т.е. тук нямаме override и не е проблем че в Human метода enjoy() е final.

14. Даден е кода:

```
class Human {
	private final void enjoy() {
		System.out.println("Enjoing...");
	}
}
public class Man extends Human {
	public final void enjoy() {
		System.out.println("Drinking beer");
	}
}
public class Test{
	public static void main(String[] args) {
		Human h;
		h = new Man();
		h.enjoy();
	}
}
```

Компилационна грешка.
Метода enjoy() на класа Human е private.

Какво ще се изведе след изпълнението на main метода?

А Enjoing...

Б Drinking beer

В Enjoing... Drinking beer

Г Компилационна грешка на `public final void enjoy() {`

Д Компилационна грешка на `h = new Man();`

Е Друга компилационна грешка

Дадена е йерархията от класове (важи за следващите 2 въпроса):

```
public abstract class Animal{
	void makeSomeNoise(){
		System.out.println("mmmm....");
	}
}

public class Dog extends Animal {
	void makeSomeNoise(){
		System.out.println("Bau bau");
	}
}

public class Cat extends Animal {
	void makeSomeNoise(){
		System.out.println("Myal myal");
	}
}

public class Bulldog extends Dog {

}
```

15. Валидни ли са следните конструкции? Подчертайте тези, които ще доведът до грешка при компилиране.

```
Animal a = new Animal();

Animal a = new Dog();

Dog d = new Animal();

Bulldog b = new Dog();

Dog d = new Bulldog();
```

Не може да се създават инстанции на абстрактен клас

Не може референция на подклас да се насочи към инстанция на базов клас. Освен това, не можем да създадем обект от тип Animal, тъй като класът е абстрактен

Не може референция на подклас да се насочи към инстанция на базов клас

16. Какво ще се изведе на конзолата? А ще се компилира ли изобщо програмата?

```
public class TestAnimal {
	public static void main(String [] args) {
		Animal animals [] = {new Dog(), new Cat(), new Cat()};
		animals[2] = new Bulldog();
		for (int i = 0; i < animals.length; i++) {
			animals[i].makeSomeNoise();
```

```
            }

      }
}
```

Благодарение на полиморфизма, тук се извикват методите характерни за типа на обекта, а не характерните за типа на референцията (Animal)

```
Bau bau
Myal myal
Bau bau
```

Класът Bulldog не предефинира метода makeSomeNoise(), а само го наследява. Затова при извикване от обект от тип Bulldog се изписва Bau bau

17. Дадено

```
class Programmer {
      Programmer debug() {
            return this;
      }
}

class JavaProgrammer extends Programmer {
      // insert code here
}
```

Кои от долу изброените конструкции могат да бъдат вместени в кода по-горе?

А `Programmer debug() { return this; }`

Б `JavaProgrammer debug() { return this; }`

В `Object debug() { return this; }`

Г `int debug() { return 1; }`

Д `int debug(int x) { return 1; }`

Е `Object debug (int x) { return this; }`

При override може да се сменя връщания тип само към наследник на връщания тип на метода на базовия клас.

Тук имаме overload, така че промяната на връщания тип не е проблем

18. Дадено

```
class X {
      void do1() {}
}

class Y extends X {
      void do2() {}
}

class Chrome {
      public static void main(String[] args) {
            X x1 = new X();
            X x2 = new Y();
            Y y1 = new Y();
            // insert code here
      }
}
```

Кои от долу изброените конструкции могат да бъдат вместени в кода по-горе, за да се компилира правилно?

А `x2.do2();`

През референция от тип X нямаме достъп до метода do2() на класа Y, въпреки че референцията сочи към обект от тип Y

Б `(Y)x2.do2();`

В `((Y)x2).do2();`

Г `x1 = x2;`

Д `x1 = y1;`

Е `y1 = (Y)x1;`

Ж Нито едно от изброените

Това няма да доведе до компилационна грешка, но по време на изпълнение, ще се хвърли ClassCastException

19. Обяснете разликите между грешки по време на компилиране и runtime грешки. Дайте по един пример за грешка от 2та типа.

Грешките по време на компилиране възникват когато имаме грешка, поради която програмата не може да се компилира и следователно изобщо не може да се стартира.
Пример за такава грешка е следният код:
int i = “Some String”;

Както говори и името, грешките по време на изпълнение възникват след като програмата е стартирана и се наричат изключения. В Java има специален механизъм за обработването им, но в най-общи линии, ако изключенията не бъдат обработени, програмата спира изпълнението си.

Например, ако се опитаме да достъпим 3-тия елемент на масив от 2 елемента, това ще доведе до ArrayIndexOutOfBoundsException.
Пример за други изключения са NullPointerException, ArithmeticException, ClassCastException, IOException.

20. Обяснете какво е полиморфизъм и каква е разликата между пренаписване (override) и презареждане (overload).

Полиморфизма е свойството на различни обекти, които имат еднакъв интерфейс и отговарят на едни и същи методи, да реагират по различен начин, в зависимост от типа на обекта. Полиморфизма е един от четирите принципа на обектно ориентираното програмиране и се получава чрез предефиниране на методи.

Предефинирането (override) е когато в производния клас, задаваме ново тяло на наследен метод на базовия клас.

Презареждане (overload) е когато дефинираме нов метод, чието име съвпада с името на друг метод на класа или на базовия клас, но се различава по списъка с аргументи.

21. Ще се компилира ли следната програма? Ако да, какво ще стане след изпълнението и?

```
public static void main(String[] args) {
      int array[] = null;
      System.out.println(array[0]);
}
```

Програмата ще се компилира, но след стартирането и, на последния ред от main метода ще се хвърли NullPointerException.

22. Даден е кода:

```
public class A {
	int x;

	A(){}

	A(int x){
		this.x = x;
	}

	String printInfo(){
		return "x is " + x;
	}
}

public class Test {

	public static void main(String[] args) {
		A array[] = new A[3];
		A a1 = new A(10);
		array[0] = a1;
		array[1] = new A();
		try {
			for (int i = 0; i < 100000; i++) {
				System.out.println(array[i].printInfo());
			}
			System.out.println("end for");
		} catch (NullPointerException e) {
			System.out.println("oops...");
		}
	}

}
```

Какво ще изведе на конзолата след извикването на main метода?

```
x is 10
X is 0
oops...
```

23. Какво не е наред със следния фрагмент?

```
public class TestException {
	Throwable cause;
	String message;

	TestException(Throwable cause) {
		this.cause = cause;
	}
}

public class A {
	int x;
	String s;

	void getClassInfo() throws TestException{
		System.out.println("x: " + x + ", s: " + s);
	}
}
```

TestException не наследява Exception

Не може след throws да стой клас, който не е наследник на Throwable

Не може след throws да стои клас, който не е наследник на Throwable.

24. Ще се компилира ли следната програма ? Ако не защо?

```
public class TestCause {

	public static void main(String[] args) {
		try {
			throw new NullPointerException("Some message.");
		} catch (NullPointerException npe) {
			throw new RuntimeException();
		}
	}
}
```

Програмата ще се компилира.

25. Даден е клас FileReader, който се компилира и е валиден. Класа има метод
`String readFile(String destinaion) throws IOException`

Какво не е наред кода по-долу? Оправете го за да може да се копилира!

```
public static void main(String[] args) {
	FileReader reader = new FileReader();
	try{
		reader.readFile("C:/test.txt");
	} catch (RuntimeException e) {
		System.out.println("Error!");
	} catch (IOException e) {
		// some code
	}
}
```

Тъй като метода `String readFile(String destinaion) throws IOException` хвърля IOException, длъжни сме или да сложим catch блок с IOException, или да упоменем че main метода хвръля IOException

26. Даден е клас FileReader, който се компилира и е валиден.
Класа има следния метод:

```
String readFile(String destinaion) throws IOException
```

Какво не е наред с кода по-долу? Оправете го за да може да се копилира!

```
public static void main(String[] args) {
	FileReader reader = new FileReader();
```

```
        try{
              reader.readFile("C:/test.txt");
        } catch (Exception e) {
              System.out.println("Error occured!");
        } catch (IOException e) {
              System.out.println("Error reading file!");
        }
}
```

IOException е наследник на Exception. В този случай,
втория catch блок ще е недостижим и това ще доведе до компилационна грешка.
Трябва обработката на IOException да става преди тази на Exception.

27. Какво ще стане ако се извика следния метод по този начин:

```
int[] m = {4, 1, 12, 5};
division(m, 2);

int[] division(int[] a, int divideBy) {

      int b[] = new int[a.length];
      for (int i = 0; i < a.length; i++) {
            b[i] = a[i] / divideBy;
            divideBy --;
      }
      return b;
}
```

При опит за делене на 0 ще се хвърли ArithmeticException
и програмата ще се прекрати

Оправете го така, че да при извикване винаги да връща масив със съответните стойности (при опит да се дели на 0 да се записва -1 в масива който се връща)

Направете го по 2 начина: посредством обработка на изключения, и без да използвате изключения.

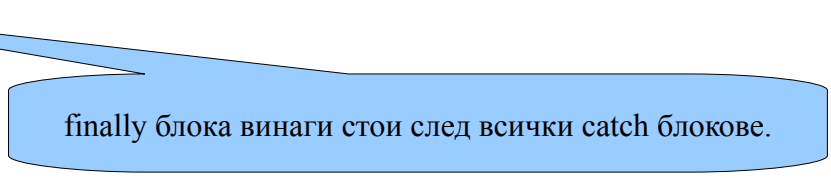
```
int[] division(int[] a, int divideBy) {

      int b[] = new int[a.length];
      for (int i = 0; i < a.length; i++) {
            if (divideBy != 0) {
                  b[i] = a[i] / divideBy;
            } else {
                  b[i] = -1;
            }
            divideBy --;
      }
      return b;
}
```

```
int[] division(int[] a, int divideBy) {

      int b[] = new int[a.length];
      for (int i = 0; i < a.length; i++) {
            try {
                  b[i] = a[i] / divideBy;
            } catch (ArithmeticException e) {
                  b[i] = -1;
            }
            divideBy --;
      }
      return b;
}
```

28. Къде е грешката?

```
public static void test(int until) {
      int x = 2;
      try {
            x *= x;
```

```
            System.out.println("X is:" + x);

        } finally {
            x = x * 100;
            System.out.println(x);
        } catch (Exception e) {
            System.out.println(x);
        }
}
```



29. Какво ще изведе следната програма:

```
public class Test {
      public static void test(int until) {
            int x = 2;
            try {
                  while(until < 100){
                        System.out.println(x);
                        x *= x;
                        if(x > until) {
                              throw new Exception();
                        }
                  }
                  System.out.println("Outside the loop!");

            } catch (Exception e) {
                  System.out.println("Error. X is: " + x);
            } finally {
                  x = x * 100;
            }
            System.out.println(x);
      }

      public static void main(String[] args) {
            test(10);
      }
}
```

```
2
4
Error. X is 16
1600
```