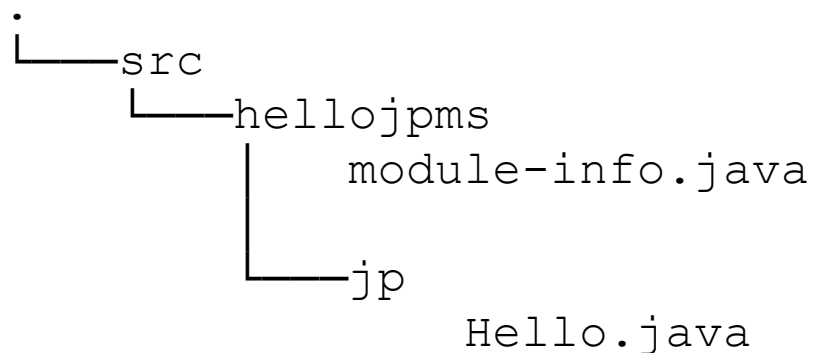


1.

Założmy, że w bieżącym katalogu (.) istnieje struktura katalogów:



z plikami o następującej zawartości:

Hello.java	module-info.java
<pre>package jp; public class Hello { public static void main(String[] args) { System.out.println("Hello"); } }</pre>	<pre>module hellojpms { }</pre>

Która z komend wydanych z konsoli systemu Windows w bieżącym katalogu pozwoli na uruchomienie klasy Hello?

- a) Żadna z wymienionych
- b) `java src\hellojpms\jp\Hello.java`
- c) `javac src\hellojpms\jp\Hello.java && java -p src -m hellojpms/jp.Hello`
- d) `javac --module-source-path src -d modules -m hellojpms && java -m hellojpms/jp.Hello`

2.

Założmy, że w bieżącym katalogu (.) istnieje struktura katalogów:

```
.
└──src
    module-info.java
    Hello.java
```

z plikami o następującej zawartości:

Hello.java	module-info.java
<pre>public class Hello { public static void main(String[] args) { System.out.println("Hello"); } }</pre>	<pre>module hellojpms { }</pre>

Która z komend wydanych z konsoli systemu Windows w bieżącym katalogu pozwoli na uruchomienie klasy Hello?

- a) Żadna z wymienionych
- b) `javac src\Hello.java src\module-info.java && java --module-path src -m /Hello`
- c) `javac src*.java && java -p src Hello`
- d) `javac --module-source-path src -d modules -m hellojpms && java -m hellojpms/ Hello`

3.

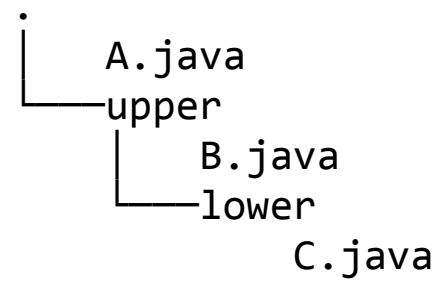
Jaki będzie wynik kompilacji poniższego kodu?

```
public class Hello {  
    {  
        Runnable r = new Runnable() { public void run() {} };  
        r = ()->{};  
    }  
  
    public static void main(String[] args) {  
        System.out.println("Hello");  
    }  
}
```

- a) Program nie skompiluje się
- b) Po kompilacji na dysku pojawi się plik Hello.class
- c) Po kompilacji na dysku pojawią się pliki Hello.class oraz Hello\$1.class
- d) Po kompilacji na dysku pojawią się pliki Hello.class, Hello\$1.class oraz Hello\$2.class

4.

Założmy, że w bieżącym katalogu (.) istnieje struktura katalogów z plikami o zawartości jak niżej.

	A.java	B.java	C.java
	<pre>package one.two.three; public class A {}</pre>	<pre>package one.two; public class B {}</pre>	<pre>package one; public class C {}</pre>

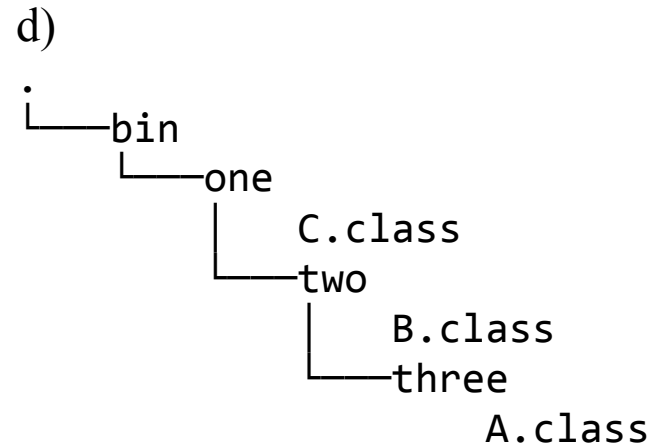
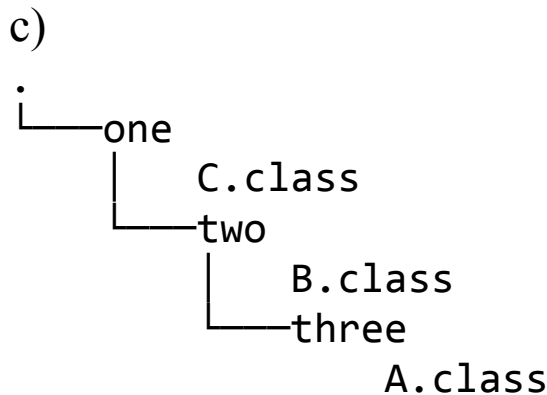
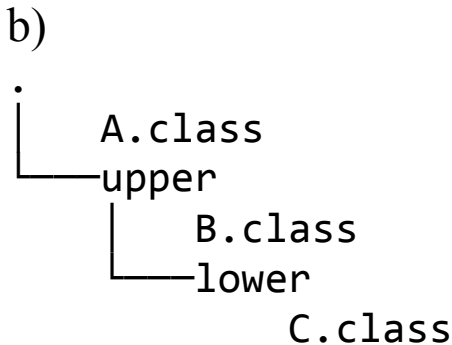
Jaki będzie efekt wywołania poniższych komend w konsoli systemu Windows

> dir /s /B *.java > bin

> javac @bin

(czy pojawi się informacja o błędzie, czy też pojawią się skompilowane klasy w katalogach jak niżej)?

a)
Pojawi się informacja o błędzie



5.

Które ze zdań są prawdziwe?

- a) Aby można było skorzystać z klasy `String` w module JPMS w jego `modules-info.java` należy wstawić deklarację **requires** `java.base`
- b) Aby można było skorzystać z klasy `String` w module JPMS w jego `modules-info.java` należy wstawić deklarację **import** `java.base`
- c) Aby można było skorzystać z klasy `String` w module JPMS w jego `modules-info.java` można wstawić deklarację **requires** `other`, gdzie `other` to nazwa modułu, którego `modules-info.java` zawiera deklarację **import transitive** `java.lang.String`
- d) Żadne z pozostałych

6.

Co można powiedzieć o poniższym kodzie?

```
public class A {  
    public static void main(String args[]) {  
        Integer i = Integer.MAX_VALUE; // 1  
        long l = 10; // 2  
        i += 1; // 3  
    }  
}
```

- a) Jego kompilacja i wykonanie przebiegnie bezbłędnie
- b) Jego kompilacja zakończy się wskazaniem błędu w linii 1
- c) Jego kompilacja zakończy się wskazaniem błędu w linii 2
- d) Jego kompilacja przebiegnie poprawnie, jednak po jego uruchomieniu podczas wykonywania linii 3 nastąpi wyrzucenie wyjątku `ArithmeticException`

7.

Co można powiedzieć o poniższym kodzie?

```
import java.util.stream.Stream;

public class A {
    public static void main(String args[]) {
        short ts1[][] = { new short[] {1,2}, {2,3}}; // 1
        short ts2[][] = new short[2][]; // 2
        ts2[0] = Stream.of(1, 2, 3).toArray(); // 3
        short[] ts3[] = new short[2][2]; // 4
    }
}
```

- a) Jest to niepoprawny kod (błąd występuje w linii 1)
- b) Jest to niepoprawny kod (błąd występuje w linii 2)
- c) Jest to niepoprawny kod (błąd występuje w linii 3)
- d) Jest to niepoprawny kod (błąd występuje w linii 4)

8.

Co można powiedzieć o poniższym kodzie?

```
interface I {  
    public void n(String s);  
}  
  
public class A implements I {  
    public var s = "s";        // 1  
    public void m() {  
        var s = "s";          // 2  
    }  
    @Override  
    public void n(var s) { // 3  
    }  
}
```

- a) Jest to niepoprawny kod (błąd występuje w linii 1)
- b) Jest to niepoprawny kod (błąd występuje w linii 2)
- c) Jest to niepoprawny kod (błąd występuje w linii 3)
- d) Jest to poprawny kod

9.

Co można powiedzieć o poniższym kodzie?

```
public class A {  
    public int i = 0;  
    public volatile int j = 0;  
    public static int k = 0;  
  
    volatile int getI() {  
        return i;  
    }  
    int getJ() {  
        return j;  
    }  
  
    synchronized int getK() {  
        return k;  
    }  
}
```

- a) Jest to niepoprawny kod (zła implementacja metody `getI()`)
- b) Jest to niepoprawny kod (zła implementacja metody `getJ()`)
- c) Jest to niepoprawny kod (zła implementacja metody `getK()`)
- d) Żadna z metod nie zapewni właściwej ochrony przy dostępie do wartości pola instancji klasy `A` w aplikacji wielowątkowej

10.

Co można powiedzieć o poniższym kodzie?

```
public class A {  
    private String name;  
  
    public void m(int i) throws Exception {  
        if (i % 3) throw new Exception("wrong value");  
    }  
  
    public A(String name) {  
        this.name = name;  
    }  
  
    public static void main(String[] args) {  
        try {  
            A a = new A("class A");  
            a.m(9);  
        } catch (Exception e) {  
            System.out.println(e.getMessage() + " in " + a.name);  
        }  
    }  
}
```

- a) Jest to niepoprawny kod (zła implementacja metody `m()`)
- b) Jest to niepoprawny kod (zła implementacja metody `main()`)
- c) Jest to poprawny kod. Po jego uruchomieniu na ekranie pojawi się `wrong value in class A`
- d) Jest to poprawny kod. Po jego uruchomieniu ekran pozostanie pusty

11.

Co można powiedzieć o poniższym kodzie ?

```
public class A {  
    public static enum Size {  
        S, M, L, X, XL, XXL  
    }  
    public static void main(String[] args) {  
        switch (Size.valueOf(args[0])) {  
            case S:  
                System.out.println("S");  
                break;  
            default:  
                System.out.println(" or not S");  
        }  
    }  
}
```

- a) Jest to poprawny kod. Po jego uruchomieniu z atrybutem S podanym w linii komend program wypisze S
- b) Jest to poprawny kod. Po jego uruchomieniu z atrybutem S podanym w linii komend program wypisze S or not S
- c) Jest to niepoprawny kod (źle zadeklarowano typ wyliczeniowy Size)
- d) Jest to niepoprawny kod (błędnie zadeklarowano opcję w instrukcji **switch**)

12.

Co można powiedzieć o poniższym kodzie?

```
import javax.swing.SwingUtilities;

public class A {
    public static void foo(String s) {           // 1
        SwingUtilities.invokeLater(new Runnable() {
            public void run() {
                System.out.print(s);
            }
        });
        s = "new";
    }

    public static void main(String[] args) {
        String s = "old";                        // 2
        foo(s);
    }
}
```

a) Jest to niepoprawny kod. Aby stał się poprawny należałoby zmienić deklarację z linii 2 na

```
final String s = "old";
```

b) Jest to poprawny kod. Po jego uruchomieniu na ekranie pojawi się new albo old

c) Jest to niepoprawny kod. Aby stał się poprawny należałoby zmienić deklarację z linii 1 na

```
public static void foo(final String s)
```

d) Jest to niepoprawny kod. Wstawienie słowa **final** w deklaracje z linii 1 lub 2 nic tu nie pomoże

13.

Co można powiedzieć o poniższym kodzie?

```
package ex00;

public class B {
    public static void m() {
        System.out.println("m1");
    }
}
```

```
import static ex00.B.m;

public class A {
    public static void m() {
        System.out.println("m2");
    }

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

- a) Jest to poprawny kod. Po uruchomieniu klasy A na ekranie wypisane zostanie m1
- b) Jest to poprawny kod. Po uruchomieniu klasy A na ekranie wypisane zostanie m2
- c) Jest to niepoprawny kod (niedopuszczalne jest pokrywanie się nazw metod importowanej i implementowanej w klasie A)
- d) Jest to niepoprawny kod (klasa A powinna być umieszczona w jakimś pakiecie, a nie jest)

14.

Co można powiedzieć o poniższym kodzie?

```
class B {  
    public void m2() {  
        throw new NullPointerException();  
    }  
  
    public void m1() {  
        try {  
            m2();  
        } catch (ArithmeticException ae) {  
            System.out.println("catch");  
        } finally {  
            System.out.println("m1()");  
        }  
    }  
}
```

```
public class A {  
    public static void main(String[] args) {  
        try {  
            B b = new B();  
            b.m1();  
            System.out.println("main()");  
        } catch (Exception e) {  
        }  
    }  
}
```

- a) Kod skompiluje się poprawnie, a po uruchomieniu klasy A na ekranie pojawi się catch
- b) Kod skompiluje się poprawnie, a po uruchomieniu klasy A na ekranie pojawi się main()
- c) Kod skompiluje się poprawnie, a po uruchomieniu klasy A na ekranie pojawi się m1()
- d) Jest to niepoprawny kod

15.
Co można powiedzieć o poniższym kodzie kodzie?

```
class E extends RuntimeException {  
    private int code;  
    public E(int code, String message) {  
        super(message);  
        this.code = code;  
    }  
    @Override  
    public String toString() {  
        return "message: " + getMessage() +  
            " code: " + code;  
    }  
}
```

```
public class A {  
    public int divide(int a, int b) {  
        if(b==0) throw new E(0, "division");  
        else return a/b;  
    }  
    public static void main(String[] args) {  
        new A().divide(0, 0);  
    }  
}
```

- a) Jest to niepoprawny kod (źle wyrzucono wyjątek)
- b) Jest to niepoprawny kod (nie przechwycono wyjątku)
- c) Jest to poprawny kod. Jednak uruchomienie klasy A zakończy się wyrzuceniem wyjątku z wiadomością rozpoczynającą się od: `Exception in thread "main" message: division code: 0`
- d) Jest to poprawny kod. Jednak uruchomienie klasy A zakończy się wyrzuceniem wyjątku z wiadomością rozpoczynającą się od: `Exception in thread "main" java.lang.RuntimeException: division`

16.

Co można powiedzieć o poniższym kodzie?

```
public class A {  
    public static void main(String[] args) {  
        int i = -1;  
        try (Scanner sc = new Scanner("012")) {  
            i = Integer.parseInt(sc.nextLine());  
        } catch (Exception e ) {  
            i = 0;  
        }  
        System.out.println(i);  
    }  
}
```

- a) Jest to poprawny kod. W wyniku uruchomienia klasy A na ekranie pojawi się 0
- b) Jest to poprawny kod. W wyniku uruchomienia klasy A na ekranie pojawi się 12
- c) Jest to poprawny kod. Jednak po uruchomieniu klasy A program zakończy działanie wyrzuceniem nieobsłużonego wyjątku
- d) Jest to niepoprawny kod

17.

Co można powiedzieć o klasie `HTTPSCClient` dodanej w `jdk11`?

- a) Podczas korzystania z tej klasy używane są metody: `getOutputStream()`, `getOutputStream()`
- b) Podczas korzystania z tej klasy używane są również klasy: `HttpRequest`, and `HttpResponse`
- c) Aby wysyłanie żądań HTTP za pomocą instancji tej klasy było możliwe należy uruchomić metodę `setDoOutput(true)`
- d) Taka klasa nie istnieje w `jdk11`

18.

Co można powiedzieć o poniższym kodzie?

```
public class A {  
  
    public <T extends Number> A(T t) {           // 1  
        // TO DO: ...  
    }  
    public A(T t) <T extends Number> {           // 2  
        // TO DO: ...  
    }  
    public static <T extends Number> A (T t) {    // 3  
        // TO DO: ...  
    }  
}
```

- a) Jest to niepoprawny kod. Błąd występuje w linii 1
- b) Jest to niepoprawny kod. Błąd występuje w linii 2
- c) Jest to niepoprawny kod. Błąd występuje w linii 3
- d) Jest to niepoprawny kod. Nie można w klasie nie-generycznej umieszczać generycznego konstruktora

19.

Co można powiedzieć o poniższym kodzie?

```
public class A {  
    private int i = 1;  
  
    static class B {  
        private int j = 0;  
  
        void m() {  
            i = 1; j = 1; // 1  
        }  
    }  
  
    public void n(int i) {  
        i += i; // 2  
    }  
  
    public static void main(String[] args) {  
        A a = new A();  
        a.n(10);  
        System.out.println(a.i);  
    }  
}
```

- a) Jest to niepoprawny kod. Błąd występuje w linii 1
- b) Jest to niepoprawny kod. Błąd występuje w linii 2
- c) Jest to poprawny kod. Po uruchomieniu klasy A na ekranie pojawi 1
- d) Jest to poprawny kod. Po uruchomieniu klasy A na ekranie pojawi 11

20.

Co można powiedzieć o poniższym kodzie?

```
import java.util.LinkedList;

public class A {
    static LinkedList<A> q = new LinkedList<>();

    public static void main(String[] args) {
        A a1 = q.poll();           // 1
        A a2 = q.remove();         // 2
    }
}
```

- a) Jest to niepoprawny kod. Błąd występuje w linii 1
- b) Jest to niepoprawny kod. Błąd występuje w linii 2
- c) Jest poprawny kod. Jednak wykonanie metody `main()` zakończy się wyrzuceniem wyjątku w linii 1
- d) Jest poprawny kod. Jednak wykonanie metody `main()` zakończy się wyrzuceniem wyjątku w linii 2

21.

Co można powiedzieć o poniższym kodzie?

```
class E<T> {}  
class B extends A {}  
class C extends B {}  
  
public class A {  
    public static <T> void m(E<? super T> e1, E<T> e2) {  
        e1 = e2;           // 1  
    }  
    public static void main(String[] args) {  
        E<A> ea=null; E<B> eb=null; E<C> ec=null;  
        A.<C>m(eb, ec);    // 2  
        A.m(eb, ea);      // 3  
    }  
}
```

- a) Jest to niepoprawny kod. Błąd występuje w linii 1
- b) Jest to niepoprawny kod. Błąd występuje w linii 2
- c) Jest to niepoprawny kod. Błąd występuje w linii 3
- d) Jest poprawny kod. Jednak wykonanie metody `main()` zakończy się wyrzuceniem wyjątku w linii 1

22.

Co można powiedzieć o poniższym kodzie?

```
interface I {
    default int n() {
        System.out.println("n()");
        return 0;
    }
}

interface J {
    default void m() {
        System.out.println("m()");
    }
}

public class A {
    public static void m(I i) {
        i.n();
    }

    public static void m(J j) {
        j.m();
    }

    public static void main(String[] args) {
        A.m(() -> {});
    }
}
```

- a) Jest to poprawny kod. W wyniku uruchomienia klasy A na ekranie pojawi się m()
- b) Jest to poprawny kod. W wyniku uruchomienia klasy A na ekranie pojawi się n()
- c) Jest to poprawny kod. W wyniku uruchomienia klasy A na ekranie nic się nie pojawi
- d) Jest to niepoprawny kod.

23.

Co można powiedzieć o poniższym kodzie (przy założeniu, że choć tego nie widać, to wszystkie importy zostały poprawnie zadeklarowane, a klasy i interfejsy są w jednym pakiecie)?

```
public class A implements I, J {  
    @Override  
    public void m() throws RemoteException { }  
  
    @Override  
    public void n() throws RemoteException { }  
  
    public static void main(String[] args) throws  
RemoteException {  
        Registry reg =  
LocateRegistry.createRegistry(3000);  
        reg.rebind("A",  
UnicastRemoteObject.exportObject(new A(), 0));  
    }  
}
```

```
interface I extends Remote {  
    public void n() throws RemoteException; }  
  
interface J extends Remote {  
    public void m() throws RemoteException; }  
  
class B {  
    public static void main(String[] args)  
throws RemoteException, NotBoundException {  
        Registry reg =  
LocateRegistry.getRegistry("localhost",  
3000);  
        Remote a = reg.lookup("A");  
        ((I) a).n(); // 1  
        ((A) a).m(); // 2  
    }  
}
```

- a) Jest to poprawny kod. Uruchomienie klasy A a następnie klasy B powiedzie się
- b) Jest to niepoprawny kod. Kompilator zgłosi błąd w linii 1
- c) Jest to niepoprawny kod. Kompilator zgłosi błąd w linii 2
- d) Jest to poprawny kod. Uruchomienie klasy A powiedzie się, jednak próba uruchomienia w następnej kolejności klasy B zakończy się wyrzuceniem wyjątku

24.

Co można powiedzieć o poniższym kodzie?

```
interface I{}  
class B extends A{}  
class C extends A{}  
class D extends B implements I{}  
class E extends D{}  
class F extends C{}  
  
public class A{  
  
    public static void main(String[] args) {  
        I i = new F(); // 1  
        A a = new E(); // 2  
        C c = new D(); // 3  
        i = new D();    // 4  
    }  
}
```

- a) Jest to niepoprawny kod. Kompilator zgłosi błąd w linii 1
- b) Jest to niepoprawny kod. Kompilator zgłosi błąd w linii 2
- c) Jest to niepoprawny kod. Kompilator zgłosi błąd w linii 3
- d) Jest to niepoprawny kod. Kompilator zgłosi błąd w linii 4

25.

Co można powiedzieć o poniższym kodzie?

```
public class A extends Thread {
    {
        Thread t = new Thread(() -> { while (true) { System.out.println("running"); } });
        t.start();
    }

    public static void main(String[] args) throws InterruptedException {
        A a = new A();
        a.start();
        a = null;
        System.gc();
        Thread.sleep(100);
    }

    @Override
    public void run() { }
}
```

a) Jest to niepoprawny kod.

b) Jest to poprawny kod. Po uruchomieniu klasy A na ekranie w nieskończoność będzie wypisywane `running`

c) Jest to poprawny kod. Po uruchomieniu klasy A na ekranie parę razy zostanie wypisane `running`

d) Jest to poprawny kod. Jednak po uruchomieniu klasy A program wyrzuci wyjątek

26.

Co można powiedzieć o poniższym kodzie?

```
public abstract class A {  
    default void m() {}  
  
    public static void main(String[] args) {  
    }  
}  
  
class B extends A {  
    void m() {  
        A.main(null);  
    }  
}
```

- a) Jest to niepoprawny kod. W klasie potomnej nie można wywoływać statycznej metody `main()` klasy nadrzędnej
- b) Jest to niepoprawny kod. W klasie potomnej nie można nadpisywać domyślnych implementacji metod odziedziczonych z klasy nadrzędnej
- c) Jest to niepoprawny kod. Klasa abstrakcyjna nie może mieć domyślnej metody
- d) Jest to poprawny kod

27.

Co można powiedzieć o poniższym kodzie?

```
public class A {  
    public static void main(String[] args) {  
        int i = 0;  
        int j = 5;  
        ext: do {  
            while (j > 0) {  
                j -= i++;  
                if (j == i)  
                    break ext;  
            }  
        } while (i < 5);  
        System.out.println(i + " " + j);  
    }  
}
```

- a) Jest to niepoprawny kod.
- b) Jest to poprawny kod. Po uruchomieniu program będzie działał w nieskończoność
- c) Jest to poprawny kod. Po uruchomieniu na ekranie pojawi się 3 3
- d) Jest to poprawny kod. Po uruchomieniu na ekranie pojawi się 5 0

28.

Co można powiedzieć o poniższym kodzie?

```
class B extends A {  
    B() { i++; }          // 1  
}  
  
public abstract class A {  
    public int i = 0;  
    A(int i) {  
        this.i = i;  
    }  
  
    public static void main(String[] args) {  
        A a = new B(); // 2  
        System.out.println(a.i);  
    }  
}
```

- a) Jest to poprawny kod. Po uruchomieniu klasy A program wypisze 0
- b) Jest to poprawny kod. Po uruchomieniu klasy A program wypisze 1
- c) Jest to niepoprawny kod. Kompilator zgłosi błąd w linii 1
- d) Jest to niepoprawny kod. Kompilator zgłosi błąd w linii 2

29.

Co można powiedzieć o poniższym kodzie?

```
public final abstract class A {  
    public static void main(String[] args) {  
    }  
}
```

- a) Jest to poprawny kod. Program się skończy zaraz po uruchomieniu klasy A
- b) Jest to poprawny kod. Jednak po uruchomieniu klasy A wyrzucony zostanie wyjątek
- c) Jest to niepoprawny kod. Klasa abstrakcyjna nie może posiadać metody `main()`
- d) Jest to niepoprawny kod. Klasa nie może być jednocześnie abstrakcyjna i finalna.

30.

Co można powiedzieć o poniższym kodzie?

```
import java.util.ArrayList;
import java.util.BoundedList;
import java.util.LinkedList;
import java.util.List;
import java.util.Vector;

public class A {
    public static void main(String[] args) {
        List l1 = new LinkedList<A>();
        List<> l2 = new ArrayList<A>();
        List<A> l3 = new BoundedList<>();
        var l4 = new Vector<A>();
    }
}
```

- a) Poprawnie zadeklarowano w nim l1
- b) Poprawnie zadeklarowano w nim l2
- c) Poprawnie zadeklarowano w nim l3
- d) Poprawnie zadeklarowano w nim l4