1.
Co można powiedzieć o poniższym kodzie?

```
interface I{}
abstract class B implements I{}
abstract class C implements I{}
class D extends B{}
public class A {
    static void m(I i) {
        if(i instanceof B) System.out.print((B) i); // 1
        else
        if(i instanceof C) System.out.print((C) i); // 2
    }
    public static void main(String[] args) {
        m(new D());                                 // 3
    }
}
```

a) Jego kompilacja przebiegnie poprawnie, jednak próba jego uruchomienia zakończy się wyrzuceniem wyjątku
b) Jego kopilacja przebiegnie poprawnie, zaś po uruchomieniu ekran pozostanie pusty
c) Jego kopilacja przebiegnie poprawnie, zaś po uruchomieniu pojawi się napis podobny do:
`D@5ca881b5`
d) Jego kompilacja nie powiedzie się z uwagi na błąd występujący w którejś z linii: `1`, `2`, `3`

2.
Co można powiedzieć o poniższym kodzie?

```
class B {}
class C extends B {}
class D extends C {}
public class A {
    static void m(B i) {
        if(i instanceof B) System.out.print((B) i+" ");   // 1
        if(i instanceof C) System.out.print((C) i+" ");   // 2
        if(i instanceof D) System.out.print((D) i+" ");   // 3
    }
    public static void main(String[] args) {
        m(new D());                                       // 4
    }
}
```

a) Jego kompilacja przebiegnie poprawnie, jednak próba jego uruchomienia zakończy się wyrzuceniem wyjątku
b) Jego kopilacja przebiegnie poprawnie, zaś po uruchomieniu pojawi się napis podobny do:
`D@5ca881b5`
c) Jego kopilacja przebiegnie poprawnie, zaś po uruchomieniu pojawi się napis podobny do:
`D@5ca881b5 D@5ca881b5 D@5ca881b5`
d) Jego kompilacja nie powiedzie się z uwagi na błąd występujący w którejś z linii: `1`, `2`, `3`, `4`

3.
Co można powiedzieć o poniższym kodzie (dla przejrzystości pominięto wymagane importy)?

```
class B {}
class C extends B {}
class D extends C {}
public class A {
    static int m(String o) {
        return switch (o.substring(1,2)) {
            case "B" -> 2;
            case "C" -> 3;
            case "D" -> 4;
            default -> 0;
        };
    }
    public static void main(String[] args) {
        System.out.print(m(new D()));
    }
}
```

a) Jest to niepoprawny kod. Jego kompilacja zakończy się błędem
b) Kod skompiluje się poprawnie, jednak po jego uruchomieniu zostanie wyrzucony wyjątek `java.lang.StringIndexOutOfBoundsException`
c) Kod skompiluje się poprawnie, a po jego uruchomieniu na ekranie zostanie wypisane `D`
d) Kod skompiluje się poprawnie, a po jego uruchomieniu na ekranie zostanie wypisane `BCD`

4.
Co można powiedzieć o poniższym kodzie (dla przejrzystości pominięto wymagane importy)?

```
enum Days{
Sun, Mon, Tue, Wed, Thu, Fri, Sat;
}
public class A {
    public static void main(String[] args) {
        Days d = Days.Mon;
        switch(d) {
            case Days.Sun, Days.Mon -> System.out.println("2");
            case Days.Mon, Days.Tue -> System.out.println("3");
            }
    }
}
```

a) Kod skompiluje się poprawnie, a po jego uruchomieniu na ekranie zostanie wypisane `2`
c) Kod skompiluje się poprawnie, a po jego uruchomieniu na ekranie zostanie wypisane `3`
c) Kod skompiluje się poprawnie, a po jego uruchomieniu na ekranie zostanie wypisane `23`
d) Jest to niepoprawny kod. Jego kompilacja zakończy się błędem

5.

```
enum Days{
Sun, Mon, Tue, Wed, Thu, Fri, Sat;
}
public class A {
    public static void main(String[] args) {
        Days d = Days.Mon;
        switch(d) {
            case Sun, Mon -> System.out.println("2");
            case Mon, Tue -> System.out.println("3");
            }
    }
}
```

a) Kod skompiluje się poprawnie, a po jego uruchomieniu na ekranie zostanie wypisane `2`
c) Kod skompiluje się poprawnie, a po jego uruchomieniu na ekranie zostanie wypisane `3`
c) Kod skompiluje się poprawnie, a po jego uruchomieniu na ekranie zostanie wypisane `23`
d) Jest to niepoprawny kod. Jego kompilacja zakończy się błędem

6.

Jeśli plik `lab02.jar` umieszczony w bieżącym katalogu ma strukturę jak niżej (z klasą `A` posiadającą metodę `main()`, pustym plikiem `MANIFEST.MF`), to które z poleceń pozwoli uruchomić klasę `A`?

`lab02.jar`
`│  module-info.class`
`│`
`├──ex01`
`│  A.class`
`│`
`└──META-INF`
`  MANIFEST.MF`
	`module-info.java`
	
	`module lab02 {}`

a) `java -jar ./lab02.jar`
b) `java -p lab02.jar lab02/ex01.A`
c) `java -cp lab02.jar ex01.A`
d) `java -cp . ex01.A`

7.
Co można powiedzieć o poniższym kodzie?

```
enum E1 { A, B, C; }
enum E2 extends { D; }
public class A {
    static void m(E1 e) {
        System.out.println(Arrays.toString(e.values()));
    }
    public static void main(String[] args) {
        m(E2.D);
    }
}
```

a) Jest to niepoprawny kod. Błędnie zadeklarowano metodę `m()`
b) Jest to niepoprawny kod. Błędnie zadeklarowano i użyto typ `E2`
c) Jest to poprawny kod. Po jego uruchomieni na ekranie zostanie wypisane `[A, B, C]`
d) Jest to poprawny kod. Po jego uruchomieni na ekranie zostanie wypisane `[A, B, C, D]`

8.
Co można powiedzieć o poniższym kodzie?

```
interface I{
    void m();
}
enum E1 implements I {
    A, B, C;
    @Override
    public void m() { System.out.print(this);}
}
public class A {
    public static void main(String[] args) {
        for(E1 e: E1.values())
            e.m();
    }
}
```

a) Jest to niepoprawny kod. Typ wyliczeniowy nie może implementować interfejsu
b) Jest to niepoprawny kod. Błędnie użyto typ `E1`
c) Jest to poprawny kod. Po jego uruchomieni na ekranie zostanie wypisane `ABC`
d) Jest to poprawny kod. Po jego uruchomieni na ekranie zostanie wypisany ciąg znaków podobny do
`E1@6af322e2E1@6af322e2E1@6af322e2`

9.

Jeśli w aplikacji JavaFX o kodzie jak niżej (pominięto wymagane importy, a klasę `Application` umieszczono w pakiecie `sample`), to jaki powinien być jej minimalny `module-info.java`?

```
public class Main extends Application {
    @Override
    public void start(Stage primaryStage) {
        try {
            StackPane root = new StackPane();
            root.getChildren().add(new Label("OK"));
            primaryStage.setScene(new Scene(root));
            primaryStage.show();
        } catch(Exception e) {}
    }
    public static void main(String[] args) {
        launch(args);
    }}
```

a)	`module sample {`
	`  requires javafx.controls;`
	`  opens sample to`
	`javafx.graphics ;`
	`}`
		b)	`module sample {`
			`  requires javafx.controls;`
			`  requires javafx.fxml;`
			`  opens sample to javafx.graphics,`
			`javafx.fxml;`
			`}`

c)	`module sample {`
	`}`
		d)	`module sample {`
			`  requires javafx.controls;`
			`}`

10.
Co można powiedzieć o poniższym kodzie (umieszczonym w jednym pliku)?

```
class B extends A {          // 1
   @Override                 // 2
   protected void m() {}
}
public class A {
   public void m() {}        // 3
}
```

a) Jest to poprawny kod
b) Jest to niepoprawny kod. Źle zadeklarowane dziedziczenie `1`
c) Jest to niepoprawny kod. Zła adnotacja w linii `2`
d) Jest to niepoprawny kod. Zły modyfikator dostępu w linii `3`

11.
Co można powiedzieć o poniższym kodzie (umieszczonym w jednym pliku)?

```
class B extends A {
    @Override                // 1
    protected void m() {} // 2
}
public class A {            // 3
    public void m() { }
}
```

a) Jest to poprawny kod
b) Jest to niepoprawny kod. Zła adnotacja w linii `1`
c) Jest to niepoprawny kod. Zły modyfikator dostępu w linii `2`
d) Jest to niepoprawny kod. Zły modyfikator dostępu w linii `3`

12.
Co można powiedzieć o poniższym kodzie?

```
final interface I { }              // 1
interface J extends I { }          // 2
public class A implements J { } // 3
```

a) Jest to poprawny kod.
b) Jest to niepoprawny kod. Błąd występuje w linii `1`
c) Jest to niepoprawny kod. Błąd występuje w linii `2`
d) Jest to niepoprawny kod. Błąd występuje w linii `3`

13.
Co można powiedzieć o poniższym kodzie (dla przejrzystości pominięto wymagane importy)?

```
public class A {
    static void m(Supplier<Integer> s) {
        System.out.println(s.get());  // 1
    }
    public static void main(String[] args) {
        A.m(x -> 0);  // 2
    }
}
```

a) Po jego skompilowaniu i uruchomieniu na ekranie pojawi się 0
b) Po jego skompilowaniu i uruchomieniu wyrzucony zostanie wyjątek
c) Jest to niepoprawny kod. Błąd wystepuje w linii `1`
d) Jest to niepoprawny kod. Błąd występuje w linii `2`

14.
Co można powiedzieć o poniższym kodzie (dla przejrzystości pominięto wymagane importy)?

```
public class A {
    interface I {
        static int f(int i) { return 2*i; }          // 1
    }
    static void m(Function<Integer,Integer> s) {
        System.out.println(s.apply(0));
    }
    public static void main(String[] args) {
        A.m(I::f);                                   // 2
    }
}
```

a) Po jego skompilowaniu i uruchomieniu na ekranie pojawi się 0
b) Po jego skompilowaniu i uruchomieniu wyrzucony zostanie wyjątek
c) Jest to niepoprawny kod. Błąd występuje w linii 1
d) Jest to niepoprawny kod. Błąd występuje w linii 2

15.
Co można powiedzieć o poniższym kodzie (dla przejrzystości pominięto wymagane importy)?

```
public class A {
    static <T> void m(Function<T, T> s, T t) {  // 1
        System.out.println(s.apply(t));
    }
    A f(A a) { return a; }
    public static void main(String[] args) {
        A a = new A();
        A.<A>m(a::f, new B());                  // 2
    }
}
```

a) Jest to niepoprawny kod. Błąd występuje w linii 1
b) Jest to niepoprawny kod. Błąd występuje w linii 2
c) Po jego skompilowaniu i uruchomieniu na ekranie pojawi się ciąg znaków podobny do `A@10f87f48`
d) Po jego skompilowaniu i uruchomieniu na ekranie pojawi się ciąg znaków podobny do `B@10f8ef40`

16.
Co można powiedzieć o poniższym kodzie?

```
System.out.println(IntStream.range(0, 5).map(x->x/2).sum());
```

a) Jest to poprawny kod. W wyniku jego wykonania na ekranie pojawi się `6`
b) Jest to poprawny kod. W wyniku jego wykonania na ekranie pojawi się `4`
c) Jest to niepoprawny kod. Źle użyto atrybutów metody `range()`
d) Jest to niepoprawny kod. Źle użyto atrybutów metoty `map()`

17.
Co można powiedzieć o poniższym kodzie (dla przejrzystości pominięto wymagane importy)?

```
public class A extends Thread{
    public void run() {
        throw new Exception("x");
    }
    public static void main(String[] args) throws Exception {
        var taba = new A[] {new A(), new A()};
        for(A a : taba)
            a.start();
    }
}
```

a) Jest to poprawny kod. Po jego uruchomieniu zostaną wyrzucone dwa wyjątki i program skończy swoje działanie
b) Jest to poprawny kod. Po jego uruchomieniu zostanie wyrzucony jeden wyjątkek i program skończy swoje działanie
c) Jest to niepoprawny kod. Źle zadeklarowano tablocę wątków w linii 2
d) Jest to niepoprawny kod. W metodzie run nie powinny pojawiać się nieobsłużone wyjątki

18.
Co można powiedzieć o poniższym kodzie?

```
class B{ void m() {} }
class C extends B{}
public class A {
    public static void main(String[] args) {
        B b = new C(); // 1
        C c = (C) b;   // 2
        c.m();         // 3
    }
}
```

a) Jego kompilacja i uruchomienie przebiegnie poprawnie.
b) Jego kompilacja przebiegnie poprawnie. Jednak podczas uruchomienia wyrzucony zostanie wyjątek w linii `1`
c) Jego kompilacja przebiegnie poprawnie. Jednak podczas uruchomienia wyrzucony zostanie wyjątek w linii `2`
d) Jego kompilacja przebiegnie poprawnie. Jednak podczas uruchomienia wyrzucony zostanie wyjątek w linii `3`

19.
Co można powiedzieć o poniższym kodzie?

```
class B{ void m() {} }
class C extends B{}
public class A {
    public static void main(String[] args) {
        C c = (C) new B(); // 1
        B b = (B) c;       // 2
        b.m();             // 3
    }
}
```

a) Jego kompilacja i uruchomienie przebiegnie poprawnie.
b) Jego kompilacja przebiegnie poprawnie. Jednak podczas uruchomienia wyrzucony zostanie wyjątek w linii `1`
c) Jego kompilacja przebiegnie poprawnie. Jednak podczas uruchomienia wyrzucony zostanie wyjątek w linii `2`
d) Jego kompilacja przebiegnie poprawnie. Jednak podczas uruchomienia wyrzucony zostanie wyjątek w linii `3`

20.
Co można powiedzieć o poniższym kodzie?

```
public class A {
    public static void main(String[] args) {
        A a = new A();
        A[] taba = new A[] { new A() };
        for(A a : taba)
            System.out.println(a);
    }
}
```

a) Jest to poprawny kod. Po jego uruchomieni na ekranie pojawi się ciąg znaków podobny do `A@10f87f48`
b) Jest to niepoprawny kod. Zduplikowano nazwę zmiennej
c) Jest to niepoprawny kod. Źle zadeklarowano tablicę
d) Jest to niepoprawny kod. Użyto złej składni przy deklaracji pętli foreach

21.
Co można powiedzieć o poniższym kodzie (dla przejrzystości pominięto wymagane importy)?

```
public class A {
    boolean m(int i) { return i>2 ? true:false; }// 1
    public static void main(String[] args) {
        var a = new A();
        if(a::m(3)) System.out.print("Y");      // 2
        System.out.println("N");
    }
}
```

a) Jest to poprawny kod. Po jego uruchomieniu na ekranie pojawi się `YN`
b) Jest to poprawny kod. Po jego uruchomieniu na ekranie pojawi się `N`
c) Jest to niepoprawny kod. Błąd występuje w linii 1
d) Jest to niepoprawny kod. Błąd występuje w linii 2

22.
Do czego służy słowo kluczowe **`native`**

a) do deklarowania klas zaimplementowanych w kodzie natywnym
b) do deklarowania metod zaimplementowanych w kodzie natywnym
c) do deklarowania lokalizacji (ustawień językowych)
d) to słowo nie jest słowem kluczowym Java

23.
Co można powiedzieć, jeśli w jakimś projekcie umieszczono następujące źródła kodu?

A.java	package-info.java

`package ex00;`
`public class A{`
`  public static void main(String[] args) {`
`  E e = E.A;`
`  }`
`}`
	`package ex00;`
	`enum E{`
	`  A, B, C;`
	`}`

a) Projekt skompiluje się poprawnie, a uruchomienie klasy `A` przebiegnie bez błędu
a) Projekt skompiluje się poprawnie, a uruchomienie klasy `A` zakończy się wyrzuceniem wyjątku
c) Projekt nie skompiluje się. Nie wolno umieszczać deklaracji typu wyliczeniowego w `package-info.java`
d) Projekt nie skompiluje się. Aby skompilował się poprawnie konieczne jest przeniesienie deklaracji typu wyliczeniowego z pliku `package-info.java` do pliku `A.java`

24.
Co można powiedzieć o poniższym kodzie?

```
public class A{
    static void m(StringBuffer sb) {
        sb = sb + "!";
    }
    public static void main(String[] args) {
        StringBuffer sb = new StringBuffer("Hello");
        m(sb);
        System.out.println(sb);
    }
}
```

a) Jest to poprawny kod. Po kompilacji i uruchomieniu klasy `A` na ekranie pojawi się napis `Hello!`
b) Jest to poprawny kod. Po kompilacji i uruchomieniu klasy `A` na ekranie pojawi się napis `Hello`
c) Jest to niepoprawny kod. Po kompilacji i uruchomieniu klasy `A` na ekranie pojawi się znak `!`
d) Jest to niepoprawny kod.

25.
Jeśli ścieżka względna do skompilowanych klas ma postać `.\bin\` oraz jeśli plik manifestu `mymanifest` znajduje się w bieżącym katalogu, to które polecenie spowoduje utworzenie pliku `library.jar` zawierającymi te klasy?

a) `jar --cf library.jar --manifest mymanifest .\bin\*`
b) `jar -cf library.jar -m mymanifest .\bin`
c) `jar -create -f library.jar --manifest mymanifest .\bin`
d) `jar --create --f library.jar --m mymanifest .\bin\*`

26.
Co można powiedzieć o poniższym fragmnence kodu (pomijając nieobsłużone wyjątki)?

```
byte[] buffer = new byte[65536];
var in = new DatagramPacket(buffer, buffer.length);
var ds = new DatagramSocket(2134);
ds.receive(in);
byte[] data = in.getData();
System.out.println(new String(data, 0, in.getLength()));
ds.close();
```

a) Jest to poprawny kod odpowiedzialny za przyjmowanie komunikatów poprzez gniazda UDP.
b) Jest to niepoprawny kod.
c) Jest to kod, po uruchomieniu którego wyrzucone zostaną wyjątki związane z przekroczeniem zakresu tablicy
d) Jest to kod, którego wykonanie jest nieblokujące (nie zależy od tego, czy przyjdzie połączenie, czy nie)

27.
Co można powiedzieć o poniższym kodzie?

```
public class A {
    int i;
    public A() throws Exception {
        i=1;
    }
    public A(int i){
        super();
        --i;
    }
    public static void main(String[] args) {
        System.out.println((new A(2)).i);
    }
}
```

a) Jego kompilacja zakończy się błędem
b) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` na ekranie zostanie wypisane `0`
c) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wypisane `1`
d) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wyrzucony wyjątek

28.
Co można powiedzieć o poniższym kodzie?

```
package ex00;

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

a) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wypisane `A`, następnie zostanie wyrzucony wyjątek `java.lang.ExceptionInInitializerError`
b) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wypisane `A`, następnie zostanie wyrzucony wyjątek `java.lang.ArithmeticException`
c) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wyrzucony wyjątek `java.lang.ExceptionInInitializerError`
d) Jego kompilacja i uruchomienie powiedzie się

29.
Co można powiedzieć o poniższym kodzie?

```
class B extends A{
}
public class A<T super B> {
    public T t;
    public static void main(String[] args) {
        A<B> a = new A<B>();
        if(a.t instanceof B)
            System.out.println("B");
    }
}
```

a) Jego kompilacja zakończy się błędem
b) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wypisane `B`
c) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` ekran pozostanie pusty
b) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wyrzucony wyjątek

30.
Co można powiedzieć o poniższym fragmencie kodu?

```
try {
    Float f = Float.parseFloat("1e0");
    byte b = f.byteValue();
    int i = f.intValue();
    double d = f.doubleValue();
    System.out.println(b + i + d);
}
catch (NumberFormatException e) {
    System.out.println("Error");
}
```

a) Jego kompilacja zakończy się błędem
b) Po jego wykonaniu na ekranie pojawi się `111.0`
c) Po jego wykonaniu na ekranie pojawi się `3.0`
d) Po jego wykonaniu na ekranie pojawi się `Error`