1.
W których liniach poniższego kodu NIE MA błędu?

```
public class A {
    { private class D{}}  // 1
    static {
        private class E{} // 2
    }
    private class F{}     // 3
    public void m()  {
        private class G{} // 4
    }
}
```

a) W linii 1
b) W linii 2
c) W linii 3
d) W linii 4

2.
Co można powiedzieć o poniższym kodzie?

```
interface I{
    public class A{
        class B{
            public class C{
                private class D{}
            };
        }
    }
}
```

a) Jest to niepoprawny kod
b) Jest to poprawny kod, ale nie da się stworzyć obiektu klasy `B` w innym kodzie
c) Jest to poprawny kod, ale nie da się stworzyć obiektu klasy `C` w innym kodzie
d) Jest to poprawny kod, ale nie da się stworzyć obiektu klasy `D` w innym kodzie

3.
Co można powiedzieć o poniższym kodzie?

```
interface I { enum E{ X, Y, Z}; }
interface J { class F{}; }
interface K { public class G{}; }

class A extends I.E {} // 1
class B extends J.F {} // 2
class C extends K.G {} // 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

4.
Co można powiedzieć o poniższym kodzie?

I.java	A.java

`package ex01;`
 	
`public interface I {`
`    int i = 10;`
`}`
	`package ex02;`
	`import ex01.I;`
	
	`public class A implements I {`
	`    {   System.out.println(i); }`
	`}`

a) W kodzie interfejsu `I` występuje błąd
b) W kodzie klasy `A` występuje błąd
c) Kod skompiluje się poprawnie, bez problemu będzie można tworzyć obiekty klasy `A` w innych klasach
d) Kod skompiluje się poprawnie, jednak przy tworzeniu obiektu klasy `A` w innych klasach wyrzucony zostanie wyjątek

5.
Co można powiedzieć o poniższym kodzie?

```
public class A {
    private A() { }

    private class G {
        private static final A a = new A();   // 1
    }
    public static A getA(){
        return G.a;                           // 2
    }
    public static void main(String[] args) {
        A a =  A.getA();                      // 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

6.
Co można powiedzieć o poniższym kodzie?

```
class A1 {
    A1(){ }          // 1
}
class A2 {
    final A2(){ }    // 2
}

class B1 extends A1{
    B1() {}          // 3
}

class B2 extends A2{
    B2() {}          // 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

7.
Jaki będzie wynik uruchomienia klasy `A` (dla przejrzystości pominięto wymagane importy)?

```
public class A {
    public static void main(String[] args) throws InterruptedException {
        Stream<Runnable> s = Stream.of(
                ()->{System.out.print("1");},
                ()->{System.out.print("2");});
        ExecutorService es = Executors.newCachedThreadPool();
        s.sequential().forEach(r -> es.execute(r));
        es.shutdown();
        es.awaitTermination(10, TimeUnit.SECONDS);
    }
}
```

a) Na ekranie zostanie wypisane `21` lub `12`
b) Na ekranie zostanie wypisane `1` lub `2`
c) Kompilacja klasy `A` zakończy się błędem
d) Uruchomienie klasy `A` zakończy się wyrzuceniem wyjątku.

8.
Która z zadeklarowanych metod jest poprawna?

```
double m1(BiConsumer<Integer, Double> c) {
    return c.accept(1);
}
double m2(BiConsumer<Integer, Double> c) {
    return c.getAsDouble(1);
}
void m3(BiConsumer<Integer, Double> c) {
    c.accept(1, 1.0);
}
void m4(BiConsumer<Integer, Double> c) {
    c.apply(1, 1.0);
}
```

a) `m1`
b) `m2`
c) `m3`
d) `m4`

9.
Co można powiedzieć o poniższym kodzie?

```
public class A {
    void m(){
        StringBuffer s1 = new StringBuffer("abcd");
        StringBuilder s2 = new StringBuilder("bcda");
        String s3 = new String("cdab");
        s3.setCharAt(0, s1.charAt(1));
        s2.setCharAt(1, s3.charAt(2));
        s1.setCharAt(2, s2.charAt(3));
        System.out.println(s1);
    }

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

a) Kod ten skompiluje się poprawnie. Po uruchomieniu klasy `A` na ekranie pojawi się `abad`
b) Kod ten skompiluje się poprawnie. Po uruchomienia klasy `A` na ekranie pojawi się `abcd`
c) Kod ten skompiluje się poprawnie. Po uruchomienia klasy `A` wyrzucony zostanie wyjątek
d) Kompilacja klasy `A` zakończy się błędem. Kompilator wskaże błąd w implementacji `m()`

10.
Które ze zdań są prawdziwe?

a) Klasę `StringBuffer` można bezpiecznie używać w aplikacjach wielowątkowych
b) Klasę `StringBuilder` można bezpiecznie używać w aplikacjach wielowątkowych
c) `StringBuffer` i `StringBuilder` mają niekompatybilne API
d) Domyślna pojemność `StringBuffer` i `StringBuilder` jest jednakowa i wynosi 32 znaki

11.
Co można powiedzieć o poniższym kodzie?

```
class B {  int i = 10; }
class D extends B {  int i = 20; }
class E extends D {  int i = 30; }

public class A<U extends B, V extends B> {
    public void m(U u, V v) {
        System.out.println(u.i + " " + v.i);
    }
    public static void main(String[] args) {
        new A().m(new D(), new E());
    }
}
```

a) Kompilacja klasy `A` zakończy się błędem
b) Kod ten skompiluje się poprawnie. Po uruchomienia klasy `A` na ekranie pojawi się `10 10`
c) Kod ten skompiluje się poprawnie. Po uruchomienia klasy `A` na ekranie pojawi się `20 30`
d) Kod ten skompiluje się poprawnie. Po uruchomienia klasy `A` na ekranie pojawi się `30 30`

12.
Co można powiedzieć o poniższym kodzie?

```
class P { int i = 10; }
class C extends P {
    int i = 20;
    static void method(P p) {
        if (p instanceof P) {
            System.out.println(((C)p).i);  // 1
        }
    }
    public static void main(String[] args) {
        C c = new C();
        C.method(c);
    }
}
```

a) Jego kompilacja zakończy się błędem w linii `1`
b) Po poprawnej kompilacji i uruchomieniu klasy `A` na ekranie pojawi się `10`
c) Po poprawnej kompilacji i uruchomieniu klasy `A` na ekranie pojawi się `20`
d) Po poprawnej kompilacji i uruchomieniu klasy `A` wyrzucony zostanie wyjątek

13.
Co można powiedzieć o poniższym kodzie?

```
record R(int i1, Integer i2){ }
class A{
    public static void main(String[] args) {
        R r1 = new R(0,0);
        R r2 = new R(Integer.parseInt("0"),Integer.parseInt("0"));   // 1
        R r3 = new R(Integer.parseInt("0"),0);                       // 2
    }
}
```

a) Jest to poprawny kod. Uruchomienie klasy `A` przebiegnie bez problemu
b) Jest to poprawny kod. Po uruchomieniu klasy `A` wyrzucony zostanie wyjątek
c) Jest to niepoprawny kod. Błąd występuje w linii `1`
d) Jest to niepoprawny kod. Błąd występuje w linii `1`

14.
Co można powiedzieć o poniższym kodzie?

```
public class A {
    public static void main(String[] args)
            throws FileNotFoundException {
        OutputStream os = new FileOutputStream("out.txt");
    }
}
```

a) Jest to niepoprawny kod (niezgodność typów podczas przypisania wartości do zmiennej `os`)
b) Jest to niepoprawny kod (nie uwzględniono wszystkich wyjątków)
c) Jest to poprawny kod, jeśli jednak na dysku nie będzie istniał plik `out.txt`, to wykonanie tego kodu zakończy się wyrzuceniem wyjątku
d) Jest to poprawny kod. Jego wykonanie przebiegnie poprawnie

15.
Załóżmy, że istnieje następująca struktura katalogów:

```
.
└───src
    └───jpms
        │   module-info.java
        │
        └───ex01
                A.java
```

z plikami o następującej zawartości:

Hello.java	module-info.java

`package ex01;`
`public class A {`
`  public static void main(String[] args) {`
`  System.out.println("A"); }`
`}`
	`module a.b.c {`
	`}`

Która z komend wydanych z konsoli systemu Windows w katalogu pokazanym przed znakiem zachęty pozwoli na uruchomienie klasy `A`?

a) `src\jpms> java .\A.java`
b) `src\jpms> javac *.java && java -p . -m a.b.c/ex01.A`
c) `src\jpms> java -p . -m a.b.c/ex01.A`
d) Żadna z wymienionych

16.
Co można powiedzieć o poniższym kodzie?

```
package ex01;

class C extends A.B {}

public class A {
    class B{}
    public static void main(String[] args) {
        A a = new A();
    }
}
```

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. Kod klasy `C` powinien pojawić się po kodzie klasy `A`
d) Jest to niepoprawny kod. Nie można rozszerzać klas wewnętrznych na zewnątrz klasy

17.
Co można powiedzieć o poniższym kodzie?

```
public class A {
    abstract class B{
    }
    class C extends A.B {
    }
    public static void main(String[] args) {
        A a = new A();
    }
}
```

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 `B` nie powinna być zadeklarowana jako abstrakcyjna
d) Jest to niepoprawny kod. Nie można rozszerzać klas wewnętrznych wewnątrz klasy

18.
Co można powiedzieć o poniższym kodzie (dla przejrzystości pominięto wymagane importy)?

```
interface I {

    static void exclamate(StringBuilder in) {
        System.out.print(in.charAt(in.length()-1)+"! ");
    }
}
public class A {
    static final String[] temp = new String[]{"A", "AB", "ABC"};
    static Random rand = new Random();
    public static void main(String[] args) {
        Stream.generate(()-> { return new StringBuilder(
            temp[rand.nextInt(3)]);})
                .forEach(I::exclamate);
    }
}
```

a) Jego kompilacja zakończy się błędem albo jego uruchomienie zakończy się wyrzuceniem wyjątku
b) Jego kompilacja powiedzie się, a po uruchomienu klasy `A` wypisane zostanie:
`A! B! C!`
c) Jego kompilacja powiedzie się, a po uruchomienu klasy `A` na ekranie pojawi się:
`A!` albo `B!` albo `C!`
d) Jego kompilacja powiedzie się, a po uruchomienu klasy `A` bez końca będą wypisywane:
`A!` albo `B!` albo `C!`

19.
Co można powiedzieć o kodzie poniższej metody (dla przejrzystości pominięto importy oraz deklarację klasy)?

```
void m() throws IOException {
    try (FileInputStream fis = new FileInputStream("A")){
        if(fis.read()==0)
        throw new ArithmeticException();
    }
}
```

a) Jego kompilacja zakończy się błędem (brak bloku `catch`)
b) Jego kompilacja zakończy się błędem (brak bloku `finally`)
c) Jego kompilacja powiedzie się, ale podczas korzystania z metody `m` trzeba obsłużyć wyjątek `IOException`
d) Jego kompilacja powiedzie się, ale podczas korzystania z metody `m` trzeba obsłużyć wyjątek `ArithmeticException`

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

```
public class A {
    static <U> void wrap(Consumer<U> function) {} // 1
    static void wrap(Runnable function) {}        // 2

    public static void main(String[] args) {
        wrap(System.out::println); // 3
        wrap(() -> {});            // 4
    }
}
```

a) Jego kompilacja zakończy się błędem w linii 1
b) Jego kompilacja zakończy się błędem w linii 2
c) Jego kompilacja zakończy się błędem w linii 3
d) Jego kompilacja zakończy się błędem w linii 4

21.
Jaki będzie wynik uruchomienia klasy o kodzie jak niżej?

```
package ex01;

class B {
    public String toString(){ return "B"; }
}
class C { }

public class A {
    public static void main(String[] args) {
        System.out.print(String.valueOf(new B())); System.out.print(" ");
        System.out.print(String.valueOf(new C())); System.out.print(" ");
        System.out.print(String.valueOf("5.2")+2);
    }
}
```

a) Na ekranie pojawi się: `B ex01.C@4c873330 5.22`
b) Na ekranie pojawi się: `B C 5.22`
c) Na ekranie pojawi się: `B ex01.C@4c873330 7.2`
d) Na ekranie pojawi się: `B C 7.2`

22.
Co można powiedzieć o poniższym kodzie?

```
class B implements Comparable<B> {
    public int i;
    public int compareTo(B o) { return o.i - i; }
}

public class A {
    public static void main(String[] args) {
        List<B> lb = new ArrayList<>();
        Collections.sort(lb);                          // 1
        Collections.sort(lb, B::compareTo);            // 2
        Collections.sort(lb, (o1, o2) -> o1.i > o2.i); // 3
    }
}
```

a) Jego kompilacja powiedzie się
b) Jego kompilacja zakończy się błędem w linii 1
c) Jego kompilacja zakończy się błędem w linii 2
d) Jego kompilacja zakończy się błędem w linii 3

23.
Co można powiedzieć o próbie uruchomienia poniższego kodu przy włączonych asercjach?

```
public class A {
    public static void main(String [] args) {
        int i = 1;
        assert (i%1 > 0)   : "negative" ;
        assert (i%1 < 0)   : "positive" ;
        System.out.println("the end");
    }
}
```

a) Po uruchomieniu klasy `A` na ekranie pojawi się napis `the end` i program skończy działanie
b) Po uruchomieniu klasy `A` wyrzucony zostanie wyjątek z odpowiednią wiadomością i na ekranie pojawi się napis `the end`
c) Po uruchomieniu klasy `A` wyrzucony zostanie wyjątek z wiadomością `java.lang.AssertionError: positive` i program skończy działanie
d) Po uruchomieniu klasy `A` wyrzucony zostanie wyjątek z wiadomością `java.lang.AssertionError: negative` i program skończy działanie

24.
Co można powiedzieć o poniższym kodzie (zakładamy, że zadeklarowano wszystkie wymagane importy)?

```
public class A {
    public static void main(String [] args) {
        List<Object> lo = Arrays.<Object>asList("A", 1, 2.2); // 1
        for (Object o : lo) {
            lo.add("A");
        }
    }
}
```

a) Po jego kompilacji i uruchomieniu wyrzucony zostanie wyjątek `StackOverflowError`
b) Po jego kompilacji i uruchomieniu wyrzucony zostanie wyjątek `ConcurrentModificationException`
c) Po jego kompilacji i uruchomieniu wyrzucony zostanie wyjątek `UnsupportedOperationException`
d) Jego kompilacja i uruchomienie przebiegnie poprawnie

25.
Co można powiedzieć o poniższym kodzie?

```
public class A {
    public static void main(String [] args) {
        int i=1;
        byte j=-128;
        System.out.println(i++ + " " + --j);
    }
}
```

a) Jest to poprawny kod. Po jego uruchomieniu na ekranie pojawi się: `2 127`
b) Jest to poprawny kod. Po jego uruchomieniu na ekranie pojawi się: `1 127`
c) Jest to poprawny kod. Po jego uruchomieniu na ekranie pojawi się: `1 -128`
d) Jest to poprawny kod. Po jego uruchomieniu na ekranie pojawi się: `2 -129`

26.
Co można powiedzieć o poniższym kodzie?

```
public static void m() throws IOException {
// ...
    HttpURLConnection url =
        (HttpURLConnection) new URL("http://pwr.edu.pl").connect(); // 1
    InputStream is = url.getInputStream();                          // 2
    OutputStream os = url.getOutputStream();                        // 3
// ...
}
```

a) Jego kompilacja powiedzie się
b) Jego kompilacja zakończy się błędem wykazanym w linii `1`
c) Jego kompilacja zakończy się błędem wykazanym w linii `2`
d) Jego kompilacja zakończy się błędem wykazanym w linii `3`

27. W którym z pakietów znajdują się definicje interfejsów wykorzystywanych w budowie aplikacji RMI?

a) `java.net`
b) `java.rmi`
c) `java.lang.rmi`
d) `java.lang.reflect`

28.
Co można powiedzieć o poniższym kodzie (zakładamy, że zadeklarowano wszystkie wymagane importy)?

```
public class A {
    public static void main(String[] args)    {
        try( Socket s = new Socket("127.0.0.256", 5000)){   // 1
            OutputStream out = s.getOutputStream();
            out.write("Hi!".getBytes());                    // 2
        } catch (Exception e) {
            e.printStackTrace();
          }
    }
}
```

a) Kompilacja tego kodu zakończy się zgłoszeniem błędu w linii `1`
b) Kompilacja tego kodu zakończy się zgłoszeniem błędu w linii `2`
c) Kod skompiluje się poprawnie, jednak podczas uruchomienia zgłoszony zostanie wyjątek
d) Kod skompiluje się i uruchomi bez błędu (pod warunkiem, że na wskazanym hoście i porcie ktoś odbierze połączenie)

29.
Co można powiedzieć o poniższym kodzie?

```
public class A {
    private static int i=1;
    private int id = i++;
    private A(){ id--; }

    public void m(A a) {
        System.out.print("1");
        if(id>2) return;
        a.m(new A());
        System.out.print("0");
    }

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

a) Jego kompilacja nie powiedzie się (nie można utworzyć obiektu klasy `A` w metodzie `main`)
b) Po jego uruchomieniu na konsoli zostanie wypisane: `1111000`
c) Po jego uruchomieniu na konsoli zostanie wypisane: `11100`
d) Po jego uruchomieniu zgłoszony zostanie wyjątek `StackOverflowError`

30.
Co można powiedzieć o poniższym kodzie?

```
public class A {
    private void m(){
        class B{                // 1
            void m(){}          // 2
        }
        new B().m();            // 3
    }

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

a) Jest to poprawny kod. Jego kompilacja i uruchomienie powiedzie się
b) Jest to niepoprawny kod. Jego kompilacja zakończy się błędem w linii `1`
c) Jest to niepoprawny kod. Jego kompilacja zakończy się błędem w linii `2`
d) Jest to niepoprawny kod. Jego kompilacja zakończy się błędem w linii `3`