1.
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

2.
Co można powiedzieć o poniższym kodzie?

```
class B {
    int i = 1;
    public B() throws Exception {
    }
}
public class A extends B {
    public A() {
        i=2;
    }
    public A(int i){
        this.i = i;
    }
    public static void main(String[] args) {
        System.out.println((new A(3)).i);
    }
}
```

a) Jego kompilacja zakończy się błędem
b) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` na ekranie zostanie wypisane 2
c) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wypisane 3
d) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wyrzucony wyjątek

3.
Co można powiedzieć o poniższym kodzie?

```
class B {
    int i = 1;
}
public class A extends B {
    final int i;

    public A(int i){
        this.i = i;
    }
    public static void main(String[] args) {
     A b = (A) new B();    // 1
     ((B) b).i = 2;        // 2
    }
}
```

a) Jego kompilacja zakończy się błędem w linii `1`
b) Jego kompilacja zakończy się błędem w linii `2`
c) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wyrzucony wyjątek `java.lang.ClassCastException`
d) Jego kompilacja i uruchomienie metody `main()` przebiegną bez błędu

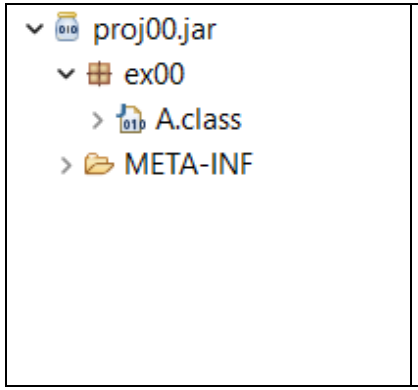
4.
Co można powiedzieć o poniższym kodzie?

```
public class A {
    static int m(int i) {
        System.out.print(i+" ");
        return i-=2;
    }
   public static void main(String[] args) {
        int i = 0;
        do {
            i ++;
            if(i%2==0) {
                continue;
            }
        } while(m(i)<5);
   }
}
```

a) Jego kompilacja zakończy się błędem
b) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` program zapętli się, zaś ekran pozostanie pusty
c) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` program zapętli się, na ekranie zaczną pojawiać się ujemne wartości
d) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` na ekranie zostanie wypisane:
`1 2 3 4 5 6 7`

5.

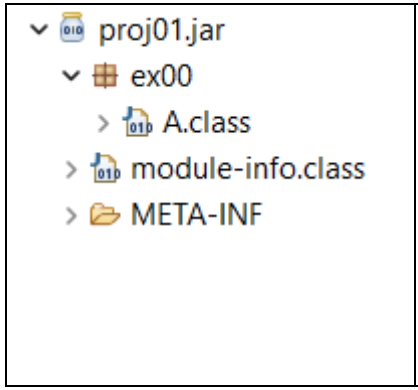
Niech `proj00.jar` będzie niemodułowym plikiem `jar` zawierającym `A.class`:




```java
package ex00;

public class A {
    public static void main(String[] args) {
        System.out.println("proj00");
    }
}
```

Niech `proj01.jar` będzie modułowym plikiem `jar` zawierającym `A.class` oraz `module-info.class`:




```java
package ex00;

public class A {
    public static void main(String[] args) {
        System.out.println("proj01");
    }
}
```

```java
module proj01 {
    exports ex00;
}
```

Co się stanie, gdy oba pliki `jar` znajdą się w tym samym katalogu, w którym wywołano komendę:

```
> java -cp proj01.jar;proj00.jar ex00.A
```

a) Na ekranie pojawi się napis: `proj00`

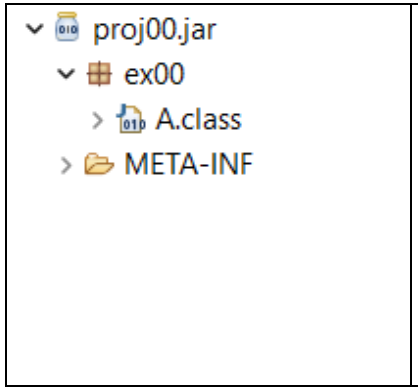
b) Na ekranie pojawi się napis: `proj01`

c) Zostanie wyrzucony wyjątek `java.lang.NoClassDefFoundError`

d) Zostanie wyrzucony wyjątek `java.lang.RuntimeException`

6.

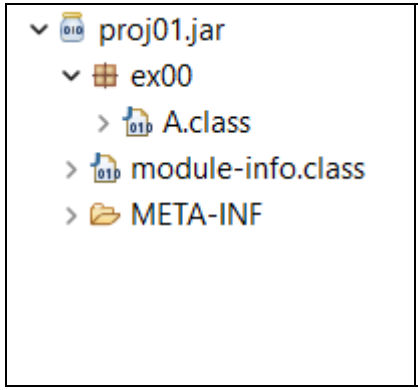
Niech `proj00.jar` będzie niemodułowym plikiem `jar` zawierającym `A.class`:




```java
package ex00;

public class A {
    public static void main(String[] args) {
        System.out.println("proj00");
    }
}
```

Niech `proj01.jar` będzie modułowym plikiem `jar` zawierającym `A.class` oraz `module-info.class`:




```java
package ex00;

public class A {
    public static void main(String[] args) {
        System.out.println("proj01");
    }
}
```

```java
module proj01 {
    exports ex00;
}
```

Co się stanie, gdy oba pliki `jar` znajdą się w tym samym katalogu, w którym wywołano komendę:

```
> java -cp proj01.jar -p proj00.jar -m proj00/ex00.A
```

a) Na ekranie pojawi się napis: `proj00`

b) Na ekranie pojawi się napis: `proj01`

c) Zostanie wyrzucony wyjątek `java.lang.NoClassDefFoundError`

d) Zostanie wyrzucony wyjątek `java.lang.module.FindException`

7.
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 zakończy się błędem
b) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wypisane `A`, następnie zostanie wyrzucony wyjątek `java.lang.ExceptionInInitializerError`
c) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wypisane `A`, następnie zostanie wyrzucony wyjątek `java.lang.ArithmeticException`
d) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wyrzucony wyjątek `java.lang.ExceptionInInitializerError`

8.
Co można powiedzieć o poniższym kodzie?

```
import static  String.*;                          // 1
public class A {
    public static void main(String[] args) {
        System.out.println(valueOf(true));     // 2
    }
}
```

a) Jego kompilacja zakończy się błędem w liniach: `1` oraz `2`
b) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wypisane `0`
c) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wypisane `true`
d) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wyrzucony wyjątek

9.
Co można powiedzieć o poniższym kodzie?

```
public class A<T> {
    public T t;

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

a) Jego kompilacja zakończy się błędem
b) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wypisane `String`
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

10.
Co można powiedzieć o poniższym kodzie?

```
public class A<T> {
    public T t;

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

a) Jego kompilacja zakończy się błędem
b) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wypisane `String`
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

11.
Co można powiedzieć o poniższym kodzie?

```
try {
    Float f = Float.parseFloat("1.0");
    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`

12.
Co można powiedzieć o poniższym kodzie?

```
public class A {
    public static class E1 extends Exception { } // 1
    public static class E2 extends E1 { }        // 2
    public static void main(String[] args) {
        try {
            throw new E2();                      // 3
        }
        catch (E1 e1) {                          // 4
            System.out.println("E1");
        }
        catch (Exception e) {
            System.out.println("E2");
        }
    }
}
```

a) Jego kompilacja zakończy się błędami w liniach `1` oraz `2`
b) Jego kompilacja zakończy się błędami w linii `3` oraz `4`
c) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` na ekranie pojawi się `E1`
d) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` na ekranie pojawi się `E2`

13.
Co można wstawić w miejsce komentarza w poniższym kodzie?

```
package ex00;

class B extends A {
    /*....*/ void m() {}
}
public class A {
    protected void m() {}
}
```

a) **public** albo nic
b) jedynie **protected**
c) **protected** albo **private**
d) **protected** albo **public**

14.
Co można powiedzieć o poniższym kodzie?

```
import java.util.stream.IntStream;

public class A {

    public static void m(Integer i) throws InterruptedException {
        if (i > 2)    throw new InterruptedException();
    }
    public static void main(String[] args) {
        try {
            IntStream.iterate(0, i -> i + 1)
              .forEach(A::m(value));
        } catch (InterruptedException e) {
                System.out.println("end");
        }
    }
}
```

a) Jego kompilacja zakończy się błędem
b) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` na ekranie w nieskończoność pojawiać się będzie `end`
c) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` na ekranie pojawi się `end`
d) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` wyrzucony zostanie wyjątek `java.lang.RuntimeException`

15.
Co można powiedzieć o poniższym kodzie?

```
class B {
}
class C extends B {
}
interface I {
    public B m();
}
interface K extends I {
    public C m();
}
```

```
public class A implements K{
    public C m() {
        return null;
    }
    public static void method(I i) {
        System.out.println((K) i.m()); // 1
    }
    public static void main(String[] args) {
        method(new A());
    }
}
```

a) Jego kompilacja zakończy się błędem z uwagi na konflikt nazw metod w deklaracji interfejsów
b) Jego kompilacja zakończy się błędem z uwagi na niedopuszczalne rzutowanie w linii `1`
c) Jego kompilacja powiedzie się, ale po uruchomieniu metody `main()` zostanie wyrzucony wyjątek
d) Jego kompilacja i uruchomienie przebiegnie bez przeszkód

16.
Co można powiedzieć o poniższym kodzie (przy założeniu, ze wszystkie wymagane importy zostały dokonane)?

```
List<Integer> li = IntStream.of(2,4,6,8,10)
        .limit(4).boxed()
        .collect(Collectors.toList());
System.out.println(li);
```

a) Jego kompilacja zakończy się błędem
b) Jego kompilacja powiedzie się, a po jego uruchomieniu na ekranie pojawi się: `[2, 4, 6, 8]`
c) Jego kompilacja powiedzie się, a po jego uruchomieniu na ekranie pojawi się: `[2, 4]`
d) Jego kompilacja powiedzie się, a po jego uruchomieniu zostanie wyrzucony wyjątek

17.
Co można powiedzieć o poniższym kodzie (przy założeniu, ze wszystkie wymagane importy zostały dokonane)?

```
public class A implements Comparable<A> {
    private final int i;
    public A(int i ){ this.i = i; }
    @Override
    public String toString() {
        return Integer.toString(i);
    }
    public static void main(String[] args) {
        TreeSet<A> tsa = new TreeSet<A>();
        tsa.add(new A(3)); tsa.add(new A(2));
        tsa.add(new A(9)); tsa.add(new A(4));
        tsa.add(new A(6)); tsa.add(new A(2));
        System.out.println(tsa);
    }
    @Override
    public int compareTo(A o) {
        return i%3==0?0:(i>o.i?1:-1); }
}
```

a) Po poprawnej kompilacji i uruchomieniu metody `main()` na ekranie pojawi się: `[2, 3, 4, 6, 9]`
b) Po poprawnej kompilacji i uruchomieniu metody `main()` na ekranie pojawi się: `[2, 2, 3, 4]`
c) Po poprawnej kompilacji i uruchomieniu metody `main()` na ekranie pojawi się: `[2, 4]`
d) Po poprawnej kompilacji i uruchomieniu metody `main()` na ekranie pojawi się: `[3, 6, 9]`

18.
Które z poniższych poleceń pozwala pozyskać informację o długości ciągu znaków w obiekcie typu `String`?

```
"A".length();      // 1
"A".length;        // 2
"A".getLenght();   // 3
"A".size();        // 4
```

a) polecenie z linii `1`
b) polecenie z linii `2`
c) polecenie z linii `3`
d) żadne

19.
Co można powiedzieć o poniższym kodzie?

```
public class A  {
    private interface I {                    // 1
        default void m() {
            System.out.println("m()");
        }
    }
    public static void main(String[] args) {
        new Thread(new I() {}::m).start(); // 2
    }
}
```

a) Jego kompilacja zakończy się błędem w linii `1`
b) Jego kompilacja zakończy się błędem w linii `2`
c) Po poprawnej kompilacji i uruchomieniu metody `main()` na ekranie pojawi się: `m()`
d) Po poprawnej kompilacji i uruchomieniu metody `main()` zostanie wyrzucony wyjątek

20.
Która z deklaracji jest błędna?

```
int ti1 [] = {1, 2, 3};
int [] ti2 = IntStream.of(1,2,3).toArray();
int [] ti3 = new int[]{1,2,3};
int ti4[3] = {};
```

a) `ti1`
b) `ti2`
c) `ti3`
d) `ti4`

21.
Niech klasa `A` o kodzie jak niżej będzie zadeklarowana w pakietach `ex01` oraz `ex02` tego samego projektu.

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

Co wtedy można powiedzieć o poniższym kodzie?

```
package ex03;
import ex01.A;
import ex02.A;

public class C {
    public static void main(String[] a) {
        A.m();
    }
}
```

a) Jego kompilacja zakończy się błędem z uwagi na kolizję nazw importowanych klas
b) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` na ekranie zostanie wypisane
`class ex01.A`
c) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` na ekranie zostanie wypisane
`class ex02.A`
d) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` zostanie wyrzucony wyjątek

22.
Co można powiedzieć o poniższym kodzie?

```
int i = 24%4;                // 1
if(!i)                       // 2
    System.out.println(1);
else
    System.out.println(2);
```

a) Jego kompilacja zakończy się błędem w linii `1`
b) Jego kompilacja zakończy się błędem w linii `2`
c) Jego kompilacja oraz wykonanie powiodą się, na ekranie pojawi się `1`
d) Jego kompilacja oraz wykonanie powiodą się, na ekranie pojawi się `2`

23.
Co można powiedzieć o poniższym kodzie?

```
int n = 0, k;
while (n > 1) k = 1;   // 1
while (false) k = 2;   // 2
while (true) k = 3;    // 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`

24.
W której linii poniższego kodu pojawi się błąd kompilacji?

```
int i1=0,i2=0;
boolean b1=true, b2=true;
switch(0) {}                // 1
switch('i') {case 'a':}     // 2
switch(i1 & i2) {}          // 3
switch(b1 && b2) {}         // 4
```

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

25.
W której linii poniższego kodu pojawi się błąd kompilacji?

```
public class A {
    enum E { One(), Two; int i = 1;};          // 1

    public static void main(String[] args) {
        switch (One.i) { case 1: break;  }     // 2
        switch (E.One.i) { case 1: break;   } // 3
        switch (A.E.One.i) { case 1: break; } // 4
     }
}
```

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

26.
Co można powiedzieć o poniższym kodzie?

```
interface I extends Remote {
    default void m() throws RemoteException { }      // 1
}
class B implements I { }                              // 2
class C extends UnicastRemoteObject implements 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

27.
Co z poniższego jest prawdą?

a) Klasa implementująca interfejs zdalny może mieć konstruktor, który nie wyrzuca wyjątku `java.rmi.RemoteException`
b) Klasa implementująca interfejs zdalny nie może dostarczać metod spoza tego interfejsu
c) Metody interfejsu zdalnego mogą zwracać jedynie wartości typów podstawowych
d) Jedynym sposobem na przekazanie namiastki do innego programu jest zarejestrowanie tej namiastki w działającym rejestrze rmi

28.
Niech kod bajtowy poniższej klasy znajduje się w bieżącym katalogu.

```
public class A {
    public static void main(String[] args) throws Exception {
        System.out.print(System.in.read());
    }
}
```

Co się stanie po wydaniu w tym katalogu komendy: `java A` , a następnie wpisaniu wartości `24 25` i naciśnięciu ENTER (przebieg sesji pokazano poniżej)

```
> java A
24 25
```

a) Na ekranie zostanie wypisane `2`
b) Na ekranie zostanie wypisane `49`
c) Na ekranie zostanie wypisane `50`
d) Na ekranie pojawi się informacja o wyrzuceniu wyjątku

29.
Co można powiedzieć o próbie kompilacji i uruchomienia metody `main()` poniższej klasy?

```
public class A {
    public static void main(String ... args) throws Exception {
        System.out.println(args[1]);
    }
}
```

a) Kompilacja zakończy się błędem
b) Kompilacja powiedzie się, a po wydaniu poniższej komendy w katalogu z kodem bajtowym:
```
> java A 2 3
```
na ekranie pojawi się 3
c) Kompilacja powiedzie się, a po wydaniu poniższej komendy w katalogu z kodem bajtowym:
```
> java -cp . A 2 3
```
na ekranie pojawi się 2
d) Kompilacja powiedzie się, a po wydaniu poniższej komendy w katalogu z kodem bajtowym:
```
> java -cp . 2 3 A
```
na ekranie pojawi się informacja o wyrzuceniu wyjątku w metodzie `main()`

30.
Co można powiedzieć o poniższym kodzie?

```
class B {
    B(){ new A(); }
}

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

a) Jego kompilacja zakończy się błędem
b) Jego kompilacja powiedzie się, a po uruchomienu metody `main()` program zakończy poprawnie swoje działanie
c) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` program będzie działał bez końca
d) Jego kompilacja powiedzie się, a po uruchomieniu metody `main()` po chwili zostanie wyrzucony wyjątek