1.
Załóżmy, że jakiś projekt eclipse posiada następującą strukturę katalogów:

```
.
├───.settings
├───bin
│   └───examples
└───src
    └───examples
```

Jaki będzie wynik wykonania poniższego programu w katalogu domowym projektu `.`
(przy założeniu, że wszystkie importy zostały prawidłowo zadeklarowane)?

```
public class A {
    public static void main(String[] args) {
        File f = new File("src");
        System.out.println(f.isDirectory());
    }
}
```

a) Program nie skompiluje się z powodu braku obsługi wyjątków
b) Program nie skompiluje się z powodu użycia nieistniejącej metody
c) Uruchomienie programu zakończy się zgłoszeniem wyjątku
d) Po uruchomieniu na ekranie pojawi się napis `true`

2.
Załóżmy, że jakiś projekt eclipse posiada następującą strukturę katalogów:

```
.
├───.settings
├───bin
│   └───examples
└───src
    └───examples
```

Jaki będzie wynik wykonania poniższego programu w katalogu domowym projektu `.`
(przy założeniu, że wszystkie importy zostały prawidłowo zadeklarowane)?

```
public class A {
    public static void main(String[] args) {
        Path p = new Path("bin");
        System.out.println(p.isDirectory());
    }
}
```

a) Program nie skompiluje się z powodu braku obsługi wyjątków
b) Program nie skompiluje się z powodu użycia nieistniejącej metody
c) Uruchomienie programu zakończy się zgłoszeniem wyjątku
d) Po uruchomieniu programu na ekranie pojawi się napis `true`

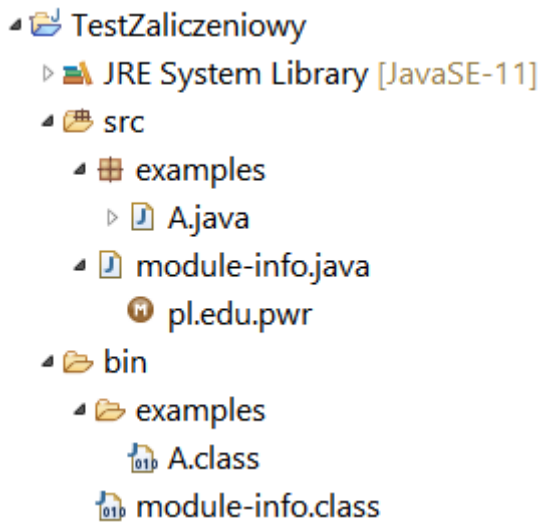
3.
Jaki będzie wynik wykonania poniższego programu
(przy założeniu, że wszystkie importy zostały prawidłowo zadeklarowane)?

```
public class A {
    static void m(int i) {
        IntStream.range(0, i)
        .forEach(System.out::print);
        }
    public static void main(String[] args) {
        IntStream.range(1, 3)
        .forEach(A::m);
    }
}
```

a) Program nie skompiluje się z powodu złej deklaracji atrybutu metody `forEach()`
b) Program uruchomi się, ale po jakimś czasie przerwie działanie z powodu przepełnienia stosu
c) Po uruchomieniu programu na ekranie pojawi się napis `001`
d) Po uruchomieniu programu na ekranie pojawi się napis `010120123`

4.

Jaką komendą wydaną z poziomu katalogu domowego projektu uruchomić klasę `A`, jeśli struktura projektu jest jak niżej?



a) `java -p bin -m pl.edu.pwr/examples.A`

b) `java -cp .\bin -m TestZaliczeniowy/examples.A`

c) `java -cp ./bin examples.A`

d) Nie da się uruchomić programu, bo kod bajtowy klasy `A` nie znajduje się w podkatalogu `bin/pl/edu/pwr/examples`

5.
Które ze zdań są prawdziwe?

a) Te same nazwy pakietów mogą pojawiać się w różnych plikach jar wykorzystywanych w tym samym projekcie, ale tylko pod warunkiem, że pliki te znajdą się one w ścieżce klas
b) Te same nazwy pakietów mogą pojawiać się w różnych plikach jar wykorzystywanych w tym samym projekcie, ale tylko pod warunkiem, że pliki te znajdą się one w ścieżce modułowej
c) W żadnym wypadku nie można używać tych samych nazw pakietów w różnych plikach jar wykorzystywanych w jednym projekcie
d) Zawsze można używać te same nazwy pakietów w różnych plikach jar wykorzystywanych w jednym projekcie

6.
Co można powiedzieć o poniższym kodzie?

```
module exhibition {
    requires transitive java.rmi; // 1
    exports communication;        // 2
    imports support;              // 3
    requires java.base;           // 4
}
```

a) Jest to poprawny kod
b) Jest to niepoprawny kod (błąd występuje w linii `1`)
c) Jest to niepoprawny kod (błądy występują w linii `2` i w linii `3`)
d) Usunięcie linii `4` nie wpłynie na poprawność bądź niepoprawność kodu

7.
Co można powiedzieć o poniższym kodzie?

```
class C extends B{}
class B extends A<C>{}                          // 1
public  class A<T> {
    public static void m(A<? super B> a) {} // 2

    public static void main(String[] args) {
     m(new A<B>());                             // 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

8.
Co można powiedzieć o poniższym kodzie?

```
public class A {
    enum E { A, B, C }

    static E m() { return A; }

    public static void main(String[] args) {
        switch(m()) {
        case A:
        case B:
        case C:
        default:
            break;
         }
    }
}
```

a) Jest to niepoprawny kod (powtórzone `A` jako nazwa klasy oraz nazwa elementu w typie wyliczeniowym)
b) Jest to niepoprawny kod (niepoprawna wartość zwracana w metodzie `m()`)
c) Jest to niepoprawny kod (źle zaimplementowano instrukcję wyboru)
d) Jest to poprawny kod

9.
Co można powiedzieć o poniższym kodzie?

```
@SuppressWarnings("serial")
class E1 extends Exception{}
@SuppressWarnings("serial")
class E2 extends E1 {}

class B extends A {
    @Override
    void m() throws E1 {
    }
}
public class A {
    void m() throws E2 {
    }
}
```

a) Jest to poprawny kod
b) Jest to niepoprawny kod (brakuje importu klasy `Exception`)
c) Jest to niepoprawny kod (metoda `m()` w klasie `B` nie może wyrzucać wyjątku `E1`)
d) Jest to niepoprawny kod (błąd zniknie po usunięciu adnotacji `@Override`)

10.
Co można powiedzieć o poniższym kodzie
(przy założeniu, że wszystkie importy zostały prawidłowo zadeklarowane)?

```
public class A {

    static Map<A,A> m = new TreeMap<>();
    public static void main(String[] args) {
        A a = new A();
        m.put(a, a);
        m.put(a, a);
        System.out.println(m);
    }
}
```

a) Jest to poprawny kod. Po jego wykonaniu na ekranie pojawi się: `{A@5594a1b5=A@5594a1b5}`
b) Jest to poprawny kod. Po jego wykonaniu na ekranie pojawi się: `{A@5594a1b5=A@5594a1b5, A@5594a1b5=A@5594a1b5}`
c) Jest to niepoprawny kod (nie można wstawać do mapy wielokrotnie tych samych obiektów)
d) Jest to niepoprawny kod (przy próbie jego wykonania pojawią się wyjątki)

11.
Co można powiedzieć o poniższym kodzie
(przy założeniu, że wszystkie importy zostały prawidłowo zadeklarowane)?

```
public class A {

    public static void main(String[] args) {
        Thread t = new Thread(
                ()->{
                    while(true)
                        System.out.println(Thread.currentThread().getName());},
                "t");
        t.start();
        t.interrupt();
    }
}
```

a) Jest to poprawny kod. Po uruchomieniu programu na ekranie w nieskończoność wypisywane będzie `t`
b) Jest to poprawny kod. Po uruchomieniu programu na ekranie wypisywane kilka razy bądź wcale `t`, po czym program zakończy się
c) Jest to niepoprawny kod (źle utworzono wątek)
d) Jest to niepoprawny kod (źle uruchomiono wątek albo źle przerwano wątek)

12.
Co można powiedzieć o poniższym kodzie
(przy założeniu, że wszystkie importy zostały prawidłowo zadeklarowane)?

```
public class A {

    public static void main(String[] args) {
        Thread t = new Thread(new Runnable() {

            @Override
            public void run() {
                System.out.println(t.getName());
            }});
        t.start();
    }
}
```

a) Jest to poprawny kod. Jednak po uruchomieniu programu zostanie wyrzucony wyjątek bo wątek nie został nazwany
b) Jest to poprawny kod. Po uruchomieniu programu na ekranie zostanie wypisywane `Thread-0`, po czym program zakończy się
c) Jest to niepoprawny kod (błąd występuje w implementacji atrybutu konstruktora wątku)
d) Jest to niepoprawny kod (błąd występuje w linii, w której wątek jest uruchamiany)

13.
Co można powiedzieć o poniższym kodzie
(przy założeniu, że wszystkie importy zostały prawidłowo zadeklarowane)?

```
class MyThread extends Thread{
    public static synchronized void  m() {
        while(true) {
            System.out.println(Thread.currentThread());
        }
    }

}
public class A {

    public static void main(String[] args) {
        Thread t1 = new Thread(MyThread::m,"t1");
        Thread t2 = new Thread(MyThread::m,"t2");
        t1.start();
        t2.start();
    }
}
```

a) Jest to poprawny kod. Uruchomienie programu spowoduje wypisywanie w nieskończoność:
`Thread[t1,5,main]`
b) Jest to poprawny kod. Uruchomienie programu spowoduje wypisywanie w nieskończoność:
`Thread[t1,5,main]` przeplatane z `Thread[t2,5,main]`
c) Jest to niepoprawny kod (źle zadeklarowano klasę `MyThread`)
d) Jest to niepoprawny kod (źle utworzono wątki albo źle je uruchomiono)

14.
Co można powiedzieć o poniższym kodzie
(przy założeniu, że wszystkie importy zostały prawidłowo zadeklarowane)?

```
public class A {

    public static void main(String[] args) throws IOException {
        ServerSocket s = new ServerSocket();        // 1
        s.bind(new SocketAddress(2140));            // 2
        s.close();                                  // 3
        s.bind(null);                               // 4
    }
}
```

a) Kod skompiluje się poprawnie, jednak podczas uruchomienia pojawi się wyjątek `SocketException: Socket is closed` przy wykonywaniu linii `4`
b) Jest to niepoprawny kod. Źle utworzono gniazd serwerowe w linii `1`
c) Jest to niepoprawny kod. Źle dowiązano adres do gniazda w linii `2`
d) Jest to niepoprawny kod. Nie można użyć `null` przy dowiązywaniu adresu do portu w linii `4`

15.
Co można powiedzieć o poniższym kodzie kodzie
(przy założeniu, że wszystkie importy zostały prawidłowo zadeklarowane,
a jego uruchomienie odbywa się z opcją `-Djava.security.manager`
przekazaną wirtualnej maszynie)?

```
public class A {

    public static void main(String[] args) throws IOException {
        ServerSocket s = new ServerSocket(2000);      // 1
    }
}
```

a) Jest to niepoprawny kod
b) Jest to poprawny kod. Jeśli port 2000 nie jest zajęty, to program się uruchomi i zaraz zamknie
c) Jest to poprawny kod. Jeśli port 2000 nie jest zajęty, to program się uruchomi i będzie czekał na przychodzące połączenia
d) Jest to poprawny kod. Jednak próba jego uruchomienia zakończy się wyrzuceniem wyjątku `AccessControlException`

16.
Co można powiedzieć o poniższym kodzie
(przy założeniu, że wszystkie importy zostały prawidłowo zadeklarowane)?

```
public class A {
    public static void main(String[] args) throws IOException {
        HttpURLConnection hc1 = new HttpURLConnection("http://pwr.edu.pl");
        HttpURLConnection hc2 = (HttpURLConnection)
URL.openConnection("http://pwr.edu.pl");
        HttpURLConnection hc3 = (HttpURLConnection) new
URL("http://pwr.edu.pl").openConnection();
        HttpURLConnection hc4 = (HttpURLConnection) new
URLConnection("http://pwr.edu.pl");
    }
}
```

a) Poprawnie utworzono `hc1`
b) Poprawnie utworzono `hc2`
c) Poprawnie utworzono `hc3`
d) Poprawnie utworzono `hc4`

17.
Co można powiedzieć o klasie `HTTPClient` 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

18.
Co można powiedzieć o poniższym kodzie (przy założeniu, że wszystkie importy zostały prawidłowo zadeklarowane, `module-info.java` jest poprawny i nie użyto żadnych parametrów JVM)?

```
 class MyException extends Exception{}

interface I extends Remote{
    void m() throws MyException;
}
public class A implements I{
    public static void main(String[] args) throws IOException, NotBoundException,
MyException {
        Registry reg = LocateRegistry.createRegistry(3000);
        A a = new A();
        reg.rebind("A", UnicastRemoteObject.exportObject(a, 0));
        System.out.println("A is ready");
        a = null;
        I i = (I) reg.lookup("A");
        i.m();
    }
    @Override
    public void m() throws MyException {
        throw new MyException();
    }
}
```

a) Kod nie skompiluje się
b) Kod skompiluje się, a po uruchomieniu w linii `2` wyrzucony zostanie wyjątek `NoSuchObjectException` lub w linii 3 `MyException` (zależnie od tego, jak szybko zadziała odśmiecanie pamięci)
c) Kod skompiluje się, a po uruchomieniu w linii `1` wyrzucony zostanie wyjątek `ExportException`
d) Kod skompiluje się, jednak zaraz po uruchomieniu pojawi się wyjątek `SecurityException`

19.
Co można powiedzieć o poniższym kodzie?

```
public class A {
    private final int i; // 1

    public void A() {     // 2
        i = 10;
    }
    public static void main(String[] args) {
        A a = new A();
    }
}
```

a) Ten kod skompiluje się i uruchomi poprawnie
b) Aby kod skompilował się i uruchomił poprawnie wystarczy usunąć słówko `final` z linii `1`
c) Aby kod skompilował się i uruchomił poprawnie należy zmienić nazwę metody w linii `2`
d) Aby kod skompilował się i uruchomił poprawnie wystarczy usunąć słówko `void` z linii `2`

20.
Co można powiedzieć o poniższym kodzie?

```
public class A {
    static void m(A a) {}
    static A a() {return new A(); }

    public static void main(String[] args) {
        var a1 = new A() {}; // 1
        A.m(a1);             // 2
        a1 = a();            // 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)