1. Co można powiedzieć o poniższym kodzie?

```
public class A {
  { i++; }
  A(){ i++; }
  static int i;

  public static void main(String[] args) {
   System.out.print(A.i);
   A a1 = new A(), a2 = new A();
   System.out.print(A.i);
  }
}
```

a) Jest to niepoprawny kod (użycie zmiennej `i` przed jej deklaracją)
b) Jest to poprawny kod. Po uruchomieniu metody `main` na ekranie zostanie wypisane `02`
c) Jest to poprawny kod. Po uruchomieniu metody `main` na ekranie zostanie wypisane `04`
d) Jest to poprawny kod. Po uruchomieniu metody `main` na ekranie zostanie wypisane `13`

2. Co można powiedzieć o poniższym kodzie?

```
@FunctionalInterface
interface I{
    void mi();
}

interface J{
    void mj();
}

public class A {
  public static void main(String[] args) {
      I i = ()->{};
      J j = ()->{};
      I ii = new I() {
          @Override
          public void mi() {}};
  }
}
```

a) Jest to niepoprawny kod (zła deklaracja jednego z interfejsów)
b) Jest to niepoprawny kod (złe użycie jednego z wyrażeń lamba)
c) Jest to niepoprawny kod (złe przypisanie do zmiennej `ii`)
d) Jest to poprawny kod.

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

```
public class A {
    public static void main(String[] args) {
      File f = new File("in.txt"); // 1
      f.createNewFile();           // 2
    }
}
```

a) Jest to niepoprawny kod (brak obsługi wyjątku w linii `1`)
b) Jest to niepoprawny kod (brak obsługi wyjątku w linii `2`)
c) Jest to poprawny kod. Jego wykonanie przebiegnie poprawnie.
d) Jest to poprawny kod, jeśli jednak na dysku będzie istniał plik `in.txt`, to wykonanie tego kodu zakończy się wyrzuceniem wyjątku (należącego do drzewa dziedziczenia `RuntimeException`)

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

```
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.

5. Co można powiedzieć o poniższym kodzie?

```
public class A {
    public <T> void h(T i) {          // 1
       System.out.println(i);
    }

   public static void main(String[] args) {
       A a = new A();
         a.h(10);          // 2
         a.h("10");        // 3
   }
}
```

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

6. Co można powiedzieć o poniższym kodzie?

```
public class A {
    public static void main(String ... args) {
      for(Object s: args) {
          System.out.print(s);
      }
    }
}
```

a) Jest to niepoprawny kod
b) Jest to poprawny kod. Zawsze zadziała, niezależnie od tego, jakie argumenty będą przekazane podczas uruchomienia.
c) Jest to poprawny kod. Jednak jego uruchomienie bez żadnych argumentów zakończy się wyrzuceniem wyjątku `java.lang.NullPointerException`
d) Jest to poprawny kod. Jednak jego uruchomienie bez żadnych argumentów zakończy się wyrzuceniem wyjątku `java.lang.ClassCastException`

7. Co można powiedzieć o poniższym kodzie?

```
import java.util.HashMap;
import java.util.Map;

public class A {
    public int i = 0;
    A(int i){ this.i = i; }

    public static void main(String[] args) {
      Map<A,Integer> map = new HashMap<A,Integer>(); // 1
      for(int i=0; i<3; i++) {
          map.put(new A(i), i);                      // 2
          System.out.println(map.get(i));
      }
    }
}
```

a) Jest to poprawny kod. Po jego uruchomieniu pojawi się: `A@15db9742 A@6d06d69c A@7852e922`
b) Jest to poprawny kod. Po jego uruchomieniu pojawi się: `null null null`
c) Jest to niepoprawny kod. Błąd występuje w linii `1`
d) Jest to niepoprawny kod. Błąd występuje w linii `2`

8. Co można powiedzieć o poniższym kodzie?

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

    public static void main(String[] args) {
        m(3.14);                    // 1
        m(new Double(2));           // 2
        for(int i=1; i<3; i++)
        m(i/10);                   // 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 static void main(String[] args) {
        i.m();                              // 1
    }
    public static I i = ()-> {};            // 2

    private interface I { void m(); } // 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

10. Co można powiedzieć o poniższym kodzie (zakładając, że zadeklarowano wszystkie importy)?

```
interface I {
    public default void n(int i) { System.out.print(i); };
}

public class A implements I {
    public void n(int i) { System.out.print(i); };

    public static void main(String[] args) {
        IntStream.iterate(0, i -> i + 2).
            limit(3).
             forEach(A()::n);
    }
}
```

a) Jest to niepoprawny kod
b) Jest to poprawny kod, po jego uruchomieniu na ekranie zostanie wypisane: `02`
c) Jest to poprawny kod, po jego uruchomieniu na ekranie zostanie wypisane: `246`
d) Jest to poprawny kod, po jego uruchomieniu na ekranie zostanie wypisane: `024`

11. Co można powiedzieć o poniższym kodzie?

```
import java.util.Arrays;
import java.util.HashSet;

public class A {
    public static void main(String[] args) {
        HashSet<String> ts = new HashSet<String>();
        ts.addAll(Arrays.asList("Ala","Ola","Ewa","Ula","Ewa","Ola"));
        System.out.println(ts);
    }
}
```

a) Jest to poprawny kod. Po wykonaniu metody `main` na ekranie pojawi się: `[Ola, Ala, Ula, Ewa]`
b) Jest to poprawny kod. Po wykonaniu metody `main` na ekranie pojawi się: `[Ala, Ewa, Ola, Ula]`
c) Jest to poprawny kod. Po wykonaniu metody `main` na ekranie pojawi się: `[Ula, Ola, Ewa, Ala]`
d) Jest to niepoprawny kod

12. Co można powiedzieć o poniższym kodzie?

```
interface I {

}
class B implements I{
}

public class A extends B {
    public void main(String[] args) {
    }
}
```

a) Jest to poprawny kod
b) Jest to niepoprawny kod (klasa `B` powinna być zadeklarowana jako klasa abstrakcyjna)
c) Jest to niepoprawny kod (metoda `main` powinna być statyczna)
d) Jest to niepoprawny kod (klasa `A` oraz klasa `B` powinny być zadeklarowane jako klasy abstrakcyjne)

13. Co można powiedzieć o poniższym kodzie?

```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

public class A {
    public static void main(String[] args) {
        ExecutorService executor = Executors.newSingleThreadExecutor(); // 1
        executor.submit(new I(){}::m);                                  // 2
    }
}
interface I {
    default void m() {
        String threadName = Thread.currentThread();                     // 3
         System.out.println("Hello " + threadName);
    }
}
```

a) Jest to niepoprawny kod (nie obsłużono wyjątku w linii 1)
b) Jest to niepoprawny kod (błędnie przypisano wartość w linii 2)
c) Jest to niepoprawny kod (przekazano zły parametr do metody w linii 3)
d) Jest to poprawny kod

14. Co można powiedzieć o poniższym kodzie?

```
interface I extends Runnable {
    default public void  run() {                    // 1
        System.out.println("OK1");
    }

}
public class A implements I {
    public static void main(String[] args) {
        Thread t1 = new Thread (new A());           // 2
        Thread t2 = new Thread (new Runnable() {    // 3
            @Override
            public void run() {
                System.out.println("OK2");
            }
        });
    }
}
```

a) Jest to niepoprawny kod (źle zadeklarowano interfejs w linii 1)
b) Jest to niepoprawny kod (źle zadeklarowano wątek w linii 2)
c) Jest to niepoprawny kod (źle zadeklarowano wątek w linii 3)
d) Jest to poprawny kod

15. Co można powiedzieć o poniższym kodzie?

```
class W extends Thread { // 1
}

public class A implements Runnable {
    public static void main(String[] args) {
        Thread t1 = new Thread(new A());
        t1 = new W();
        t1.run();
    }

    @Override
    public void run() {
        System.out.println("OK");
    }
}
```

a) Jest to niepoprawny kod (źle zadeklarowano klasę `W`)
b) Jest to poprawny kod. Po wykonaniu metody `main` na ekran nie pojawią się żadne napisy
c) Jest to poprawny kod. Po wykonaniu metody `main` na ekranie pojawi się napis `OK`
d) Jest to poprawny kod, jednak w trakcie wykonywania metody `main` zostanie wyrzucony wyjątek

16. Co można powiedzieć o poniższym kodzie?

```
class E1 extends Exception { super(); }

interface I {
    public void m1() ;
    public void m2() throws E1;
}

public class A implements I {
    @Override
    default public void m1() {} // 1
    @Override
    public void m2() {}         // 2
}
```

a) Jest to poprawny kod.
b) Jest to niepoprawny kod (źle zaimplementowano metodę `m1` w linii `1`)
c) Jest to niepoprawny kod (źle zaimplementowano metodę `m2` w linii `2`)
d) Jest to niepoprawny kod (źle zadeklarowano klasę `E1`)

17. Co można powiedzieć o poniższym kodzie?

`package p2;`
 	
`public class B {`
`}`
	`package p2;`
	
	`public class A {`
	`  static public class B { }`
	`  public static void main(String[] args) {`
	`  System.out.println(new B());`
	`  }`
	`}`

a) Jest to poprawny kod. Po wykonaniu metody `main` na ekranie pojawi się: `p2.B@15dbaa42`
b) Jest to poprawny kod. Po wykonaniu metody `main` na ekranie pojawi się: `p2.A$B@15db9742`
c) Jest to niepoprawny kod (nie można klas z tego samego pakietu deklarować w osobnych plikach)
d) Jest to niepoprawny kod (nie można do `println` wstawić obiektu utworzonego operatorem `new`)

18. Co można powiedzieć o poniższym kodzie?

```
public abstract class A {
    class B extends A {
        void getC() { }
    }
    public String c = "A";
    private abstract void getC() ;    // 1
}
```

a) Jest to poprawny kod, jednak nie da się utworzyć w innym źródle kodu instancji klasy `B` i wywołać jej metody `getc()`
b) Jest to poprawny kod, można utworzyć w innym źródle kodu instancję klasy `B` i wywołać jej metody `getC()`
c) Jest to niepoprawny kod (błąd występuje w linii `1`)
d) Jest to niepoprawny kod (klasa abstrakcyjna nie może mieć klas wewnętrznych)

19. Co można powiedzieć o poniższym kodzie?

```
Thread t1 = new Thread(new Runnable() {
   public void run() {
      try {
         wait();
      } catch (InterruptedException e) {
         e.printStackTrace();
      }
   }
});
Thread t2 = new Thread(new Runnable() {
   public void run() {
      notify();
   }
});
t2.start();
t1.start();
```

a) Po jego uruchomieniu wątek `t1` nie zakończy działania
b) Po jego uruchomieniu oba wątki zakończą swoje działanie
c) Po jego uruchomieniu zgłoszony zostanie wyjątek `IllegalMonitorStateException`
d) Jego kompilacja zakończy się błędem

20. Co można powiedzieć o poniższym kodzie?

```
public class A {
    private  int i ;
     public void m1() { i++;}
     public void m2() { i--;}

     public static void main(String[] args) {
        A a = new A();
        try {
            a.m1();
        } catch(Exception e) {}
        finally {
            a.m2();
        }
    }
}
```

a) Jest to poprawny kod
b) Jest to niepoprawny kod (brak dostępu do zmiennej w bloku `finally`)
c) Jest to niepoprawny kod (nie można zakładać bloku try-catch na fragmencie kodu, w którym nie wyrzuca się żadnych wyjątków)
d) Jest to niepoprawny kod (nie można deklarować pola `i` jak wyżej bez jego zainicjalizowania)

21. Co można powiedzieć o poniższym kodzie?

```
class B extends A {
    B(){ i++;}
}
class D extends B {
    D() { i+=2;}
}

public class A {
    int i;
    int getI() {return i;}
    public static void main(String[] args) {
        System.out.println(
                new A().getI() + "" +
                new B().getI() + "" +
                new D().getI());
    }
}
```

a) Jest to poprawny kod. Po wykonaniu metody `main` na ekranie pojawi się: `013`
b) Jest to poprawny kod. Po wykonaniu metody `main` na ekranie pojawi się: `012`
c) Jest to poprawny kod. Po wykonaniu metody `main` na ekranie pojawi się: `001`
d) Jest to niepoprawny kod (brak dostępu do pola `i` w klasach `B` i `D`)

22. Co można powiedzieć o poniższym kodzie?

```
public class A {
    public static int m(int i) throws Exception{
        if (i<=0) throw new Exception("Exception ");
        System.out.print("No exception ");
        return i++;
    }
    public static void main(String[] args) {
        int i = 10;
        try {
            i=m(0);
        } catch (Exception e) {
            System.out.print(e.getMessage());
        } finally {
            System.out.print("Finally ");
        }
        System.out.println(i);
    }
}
```

a) Jest to poprawny kod. Po wykonaniu metody `main` na ekranie pojawi się: `No exception Finally 0`
b) Jest to poprawny kod. Po wykonaniu metody `main` na ekranie pojawi się: `No exception Finally 1`
c) Jest to poprawny kod. Po wykonaniu metody `main` na ekranie pojawi się: `Exception Finally 10`
d) Jest to niepoprawny kod

23. Co można powiedzieć o poniższym kodzie?

```
public class A {
    final StringBuffer s = new StringBuffer("A");

    public StringBuffer getS() {
        return s;
    }
   public static void main(String[] args)  {
        A a = new A();
        StringBuffer as = a.getS();        // 1
        as.append("....");                 // 2
        as = new StringBuffer(as);         // 3
        System.out.println(as);
   }
}
```

a) Jest to niepoprawny kod (błędne przypisanie w linii `1`)

b) Jest to niepoprawny kod (zmienna lokalna stała się zmienną finalną, więc nie można użyć `append` w linii `2`)

c) Jest to niepoprawny kod (błędne przypisanie wartości do zmiennej, która stała się zmienną finalną w linii `3`)

d) Jest to poprawny kod. Po jego uruchomieniu na ekranie pojawi się `A....`

24. Co można powiedzieć o poniższym kodzie?

```
int i=1;
byte j=127;
i =+ j++;
System.out.println(i+ " " + j);
```

a) Jest to niepoprawny kod
b) Jest to poprawny kod. Po jego uruchomieniu zostanie zgłoszony wyjątek
c) Jest to poprawny kod. Po jego uruchomieniu na ekranie pojawi się: `127 -128`
d) Jest to poprawny kod. Po jego uruchomieniu na ekranie pojawi się: `128 -127`

25. Która deklaracja jest poprawna?

```
Long l1 = new Long(20); // 1
Long l2 = 3000000;      // 2
Double d1 = 40;         // 3
Double d2 = 1E-10.0;    // 4
```

a) Deklaracja w linii 1
b) Deklaracja w linii 2
c) Deklaracja w linii 3
d) Deklaracja w linii 4

26. Co można powiedzieć o poniższym fragmencie kodu?

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

a) Jest to poprawny fragment kodu.
b) Jest to niepoprawny fragment kodu (klasa `URL` nie dysponuje metodą `getInputStream()`)
c) Jest to niepoprawny fragment kodu (nie obsłużono wszystkich wyjątków)
d) Jest to niepoprawny kod (złe wywołanie konstruktora klasy `URL`)

27. Co można powiedzieć o poniższym kodzie (zakładając, że wszystkie importy zostały poprawnie zadeklarowane)?

```
public class A extends UnicastRemoteObject  {
    protected A() throws RemoteException {}
    @Override
    public void m(int i) {
        System.out.println(i);
    }

    public static void main(String[] a)
            throws Exception{

        LocateRegistry.
          createRegistry(2000).
           bind("A", new A());
    }
}
```

```
public interface I extends Remote{
   public void m(int i)
     throws RemoteException;
}
```

```
public class B {
public static void main(String[] a)
            throws Exception {

 ((I) LocateRegistry.
        getRegistry(2000).
         lookup("A")).m(10);
           }
}
```

a) Jest to poprawny kod. Po uruchomieniu w różnych konsolach, kolejno, klasy `A` i klasy `B`, na konsoli klasy `B` pojawi się `10`

b) Jest to poprawny kod. Po uruchomieniu w różnych konsolach, kolejno, klasy `A` i klasy `B`, na konsoli klasy `A` pojawi się `10`

c) Jest to poprawny kod. Jednak po uruchomieniu w różnych konsolach, kolejno, klasy `A` i klasy `B`, na konsoli klasy `B` zostanie zgłoszony wyjątek (gdyż klasa `A` zbyt szybko zakończy swoje działanie)

d) Jest to niepoprawny kod

28. Co można powiedzieć o poniższym kodzie?

```
public class A {
    @Override
   protected void finalize() throws Throwable {
       B.a = this;
   }
   public static void main(String[] args) throws InterruptedException  {
        B.a = new A();
        B.a = null;
        System.gc();
        Thread.currentThread().sleep(1000);
        System.out.print(B.a + " ");
        B.a = null;
        System.gc();
        Thread.currentThread().sleep(1000);
        System.out.println(B.a);
   }
}
```

a) Jest to poprawny kod. Po uruchomieniu na ekranie pojawi się `A@15db9742 null`

b) Jest to poprawny kod. Po uruchomieniu na ekranie pojawi się `A@15db9742 A@15db9742`

c) Jest to poprawny kod. Po uruchomieniu na ekranie pojawi się `null null`

d) Jest to niepoprawny kod

29. Jeśli struktura katalogów jest jak niżej, to które z poleceń pozwoli uruchomić z linii komend klasę `A` (przed znakiem zachęty pokazano bieżącą ścieżkę)?

<table>
<tr>
<td>
<pre>
.
├──bin
│ └──p1
│   └──p2
│   A.class
│
└──src
   └──p1
      └──p2
         A.java
</pre>
</td>
<td>
<pre>
package p1.p2;

public class A {
    public static void main(String[] args) {
    }
}
</pre>
</td>
</tr>
</table>

a) `.\bin\p1> java -cp .\..\.. p1.p2.A`
b) `.\bin\p1\p2>  java -cp . A`
c) `.> java -cp .\bin p1.p2.A`
d) `.\src\p1\p2> java -cp .\..\.. p1.p2.A`

30. Co można powiedzieć o poniższym kodzie (zakładając, że wszystkie konieczne importy zostały wykonane)?

```
class A {
    public class B {
        private int i;
    }
    public static void main(String[] args) {
     Socket so;
     try {
        so = new Socket("192.168.1.200", 2000); // 1
        ObjectInputStream out = new ObjectInputStream(so.getInputStream());
        B b = (B) out.readObject();             // 2
        System.out.println(b.i);                // 3
     } catch (Exception e) {}
    }
}
```

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)