```java
// A1.java

public class A1 {
	private A1 a1;
	static{
		System.out.println("A1-S");
	}
	static{
		System.out.println("A1-S2");
	}
	{
		System.out.println("A1-N");
	}
	public A1() {
		System.out.println("A1");
	}
	public A1(A1 a1){
		System.out.println("A1(A1)");
		this.a1 = a1;
	}
	void metoda(){
		System.out.println("metoda A1");
	}
	public static void main(String[] args) {
		A4 a4 = new A4();
		a4.metoda();
		a4.metoda2();
		new A2(a4);
	}
}

class A2 extends A1 {
	A1 a1;
	public A2() {
		this(new A1());
		System.out.println("A2");
	}
	public A2(A1 a1){
		this.a1 = a1;
		System.out.println("A2(A1)");
	}
	public void metoda(){
		System.out.println("metoda A2");
	}
	private void metoda2(){
		System.out.println("metoda2 A2");
	}
}

class A3 extends A2 implements Serializable {
	public A3() {
		System.out.println("A3");
	}
	public A3(A2 a2) {
		this();
		System.out.println("A3(A2)");
	}
	public A3(A2 a2, A1 a1) {
		this(a2);
		System.out.println("A3(A2, A1)");
	}
	protected void metoda(){
		System.out.println("metoda A3");
	}
	public void metoda2(){
		System.out.println("metoda2 A3");
	}
}
```

```
class A4 extends A3 {
       private A1 a = new A2();
       private A2 a2 = new A2(new A1(new A1()));
       Serializable a3 = new A3(a2, a1);
       public A4() {
              a2.metoda();
              System.out.println("A4");
              a.metoda();
              ((A1) a3).metoda();
       }
       protected void metoda(){
              System.out.println("metoda A4");
       }
}
```

```java
// C1.java

public class C1 {
        C1() {
                System.out.println("C1");
        }

        public static void main(String[] args) throws CE1, Exception {
                C1 c1 = new C1();
                C2 c2 = new C2();
                try {
                        System.out.println(c2.metoda(c2));
                } catch (CE2 e) {
                        System.out.println("main 2: " + e);
                } catch (CE1 e) {
                        System.out.println("main 3: " + e);
                } catch (Error e) {
                        System.out.println("main 4: " + e);
                } catch (Throwable e) {
                        System.out.println("main 5: " + e);
                } finally {
                        System.out.println("finally");
                }
                System.out.println(c1.metoda(c2));
                new C1().metoda(new C1());
                c2.close();
        }

        private boolean t = false;

        Object metoda(C1 c) throws CE1 {
                if(t)
                        t = false;
                else
                        t = true;
                if (c instanceof C1 && (t)) {
                        System.out.println("method");
                } else {
                        throw new CE2();
                }
                return 1;
        }
}

class C2 extends C1 implements AutoCloseable {
        C2() {
                System.out.println("C2");
        }
        Object metoda(C1 c) throws CE1 {
                System.out.println("method C2");
                if (errorCheck() && c instanceof C2)
                        throw new CE2("Error 2");
                else if (c instanceof C2)
                        throw new CE1();
                else
                        return new String("abc");
        }

        boolean errorCheck() throws CE1 {
                return metoda(null)!=null;
        }
        @Override
        public void close() throws Exception {
                System.out.println("C2: close()");
        }
}
```

```
class CE1 extends Throwable {
	public CE1() {
		System.out.println("CE1 - 1");
	}
	public CE1(String s) {
		super(s);
		System.out.println("CE1 - 2");
	}
}

class CE2 extends RuntimeException {
	public CE2() {
		System.out.println("CE2 - 1");
	}
	public CE2(String s) {
		this();
		System.out.println("CE2 - 2");
	}
}


// D1.java

public class D1 {

	public static void main(String[] args) {
		String test = "FEE Banjaluka!";
		String res = method(t -> {
			String result = "";
			for (int i = t.length() - 1; i >= 0; i--) {
				result += t.charAt(--i);
			}
			return result;
		}, test);
		System.out.println(res);
		res = method(new D2()::method2, test);
		System.out.println(res);
		res = new DI() {
			public String exec(String s) {
				return s.toLowerCase();
			}
		}.exec(test, 4);
		System.out.println(res);
	}

	static String method(DI sf, String s) {
		return sf.exec(s);
	}
}

class D2 {
	public String method2(String s) {
		String result = "";
		for (int i = s.length() - 1; i >= 0; i--) {
			result += s.charAt(i);
		}
		return result;
	}
}

interface DI {
	public String exec(String s);
	default String exec(String s, int i) {
		return s.substring(i);
	}
}
```

```java
// E1.java

public class E1 extends Thread {
	String name;
	public E1(String name) {
		setDaemon(true);
		this.name = name;
	}
	public void run() {
		new Thread(new Runnable() {
			public void run() {
				for(int i=0; i<3; i++){
					System.out.println(name + "1: " + i);
				}
			}
		}).start();
		new Thread(){
			public void run(){
				for (int i = 0; i < 3; i++){
					System.out.println(name + "2: " + i);
				}
			}
		}.run();
		new Thread() {
			void Thread(){
				start();
			}
			public void run() {
				try {
					System.out.println(name + "3: --------");
					this.join();
				} catch (InterruptedException e) {
					e.printStackTrace();
				}
				for (int i = 0; i < 3; i++) {
					System.out.println(name + "3: " + i);
				}
			}
		}.start();
		System.out.println(name);
	}

	public static void main(String args[]){
		System.out.println("START");
		E1 a = new E1("A");
		E2 b = new E2("B");
		E3 c = new E3("C", a);
		E3 d = new E3("D", a);
		a.start();
		b.start();
		try {
			a.join();
			b.join();
		} catch (InterruptedException e) {
			e.printStackTrace();
		}
		System.out.println("END");
	}
}

class E2 extends E1 implements Runnable{

	public E2(String name) {
		super(name);
	}

}
```

```
class E3 extends E2{
        Thread t;
        public E3(String name, Thread t) {
                super(name);
                this.t = t;
                start();
        }
        public void run(){
                try {
                        t.join();
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }
                for(int i=0; i<3; i++){
                        System.out.println(name + "1: " + i);
                }
        }
}
```

```
package org.unibl.etf.assignment01;

public class A {
        B b1 = new B();
        B b2 = new B();
        static {
                System.out.println("static A");
        }
        {
                System.out.println("non-static A");
        }
        public A() {
                System.out.println("Constructor A");
        }
        public static void main(String[] args) {
                new A();
                new B();
        }
}

class B {
        static {
                System.out.println("static B");
        }
        {
                System.out.println("non-static B");
        }
        public B() {
                System.out.println("Constructor B");
        }
}
```

redoslijed korištenja instanci
klasa određuje redoslijed
poziva statičkih blokova

1

[✓ A]	B
static A static B non-static B Constructor B non-static B Constructor B non-static A Constructor A non-static B Constructor B	static A non-static A Constructor A static B non-static B Constructor B non-static B Constructor B non-static B Constructor B
C	**D**
static A static B non-static A Constructor A non-static B Constructor B non-static B Constructor B non-static B Constructor B	static A Constructor A non-static A static B Constructor B non-static B Constructor B non-static B Constructor B non-static B

```java
package org.unibl.etf.assignment02;

public class A {
	static {
		int x = 5;
	}

	static int x, y;

	public static void main(String args[]) {
		x--;
		System.out.println(x);
		System.out.println(y);
		metoda();
		System.out.println(x);
		System.out.println(y);
		System.out.println(++x + x++);		// B
		System.out.println(++A.x);		// C
	}

	public static void metoda() {
		y = ++x;
	}
```

2

A 4 0 5 5 12 8	**B** Greška pri kompajliranju u liniji koda označenoj oznakom B
C Greška pri kompajliranju u liniji koda označenoj oznakom C	[✓ **D**] -1 0 0 0 2 3

```
package org.unibl.etf.assignment03;

public class A {

        static double i = 1;
        static int j = 2;
        int x = 3;
        static int y = 6;

        public static void main(String args[]){
                metoda();
                System.out.println(i + j);
                System.out.println(x + i);        // B
                metoda2();
                System.out.println(i + y);
                System.out.println(i + j);
        }

        public static int metoda(){
                return (int)i + --y + (j++);
        }

        public static double metoda2(){
                return  j++ - --i;
        }
};
;;                                                  // D
```

promjenljiva x je vezana za instancu, nije statička promjenljiva

3	
A 4.0 4.0 5.0 4.0	[✓ B] Greška pri kompajliranju u liniji koda označenoj oznakom B
C 4 4 5 4	D Greška pri kompajliranju u liniji koda označenoj oznakom D

```
package org.unibl.etf.assignment04;

public class A {
        static int x = 3;                               // C
        public static void main(String[] args) {
                new A();
        }

        A() {
                A(2);           this(2) ✓               // D
        }

        A(int x) {
                System.out.println(x);
        }
}
```

4	
A 2	**B** 3
C Greška pri kompajliranju u liniji koda označenoj oznakom C	[✓ **D**] Greška pri kompajliranju u liniji koda označenoj oznakom D

```
package org.unibl.etf.assignment05.a;

public class A {

        private void methodA() {
        }

        void methodB() {            package-private
        }
                                       unutar klase, podklasa čak i van
        protected void methodC() {     paketa i unutar istog paketa
        }

        public void methodD() {
        }
}


package org.unibl.etf.assignment05.b;

import org.unibl.etf.assignment05.a.A;

public class Main {

        public static void main(String[] args) {
                A a = new A();
                a.methodA();                // A
                a.methodB();                // B
                a.methodC();                // C
                a.methodD();                // D
        }

}
```

5

A Greška pri kompajliranju u liniji koda označenoj oznakom A	**B** Greška pri kompajliranju u linijama koda označenim oznakama A i B
[✓ **C**] Greška pri kompajliranju u linijama koda označenim oznakama A, B i C	**D** Greška pri kompajliranju u linijama koda označenim oznakama A, B, C i D

```
package org.unibl.etf.assignment07;

public class A {
        int 1abc;
        int abc_1;
        int oneAbc;
        int final;
        int $while;
}
```

while$ takođe validan

6	
A Svi identifikatori su validni	B Svi identifikatori, osim $while, su validni
[✓ C] Validni identifikatori su: abc_1, oneAbc, $while	D Validni identifikatori su: abc_1, oneAbc
E Validan je samo identifikator oneAbc	F Nema validnih identifikatora

```
package org.unibl.etf.assignment07;

public class B {
        public static void main(String[] args) {
                int i = 1;
                int n = ++i % 5;
                System.out.print(n);
                n = i-- % 4;
                System.out.print(n);
                n = i++ % 2;
                System.out.print(n);
        }
}
```

7	
[✓ A] ✓ 221	B 110
C 121	D 220
E Izuzetak pri izvršavanju	F Greška pri kompajliranju

```
package org.unibl.etf.assignment07;

public class C {
        public static void main(String arg[]) {
                int i = 4;
                for (; i <= 12; i+=3) {
                        i = i++;
                        i -= 1;
                        i++;
                        i += 1;
                        i = i++;
                }
                System.out.println(--i);
        }
}
```

8	
[✓ A] ✓ 15	B 18
C 17	D 13
E 16	F 14

```
package org.unibl.etf.assignment07;

public class D {
        String outerProperty = "1";

        void method() {
                N nestedClass = new N();
                nestedClass.method();
        }

        public class N{
                String innerProperty = "2";
                void method() {
                        System.out.print(outerProperty + innerProperty);
                }
        }

        public static void main(String[] args) {
                D outerClass = new D();
                outerClass.method();
                D.N nested = outerClass.new N();
                nested.method();
        }
}
```

9

[✓ **A**] Ispis 1212	**B** Ispis 12, pa izuzetak pri izvršavanju
C Greška pri kompajliranju	**D** Izuzetak pri izvršavanju

```
package org.unibl.etf.assignment08;


class MyError extends Error {
}

public class A {
	public static void main(String args[]) {
		try {
			test();
		} catch (Error ie) {
			System.out.println("Error caught");
		}
	}

	static void test() throws Error {
		throw new MyError();
	}
}
```

10

A Izuzetak pri izvršavanju: metoda test ne može baciti MyError	**B** Greška pri kompajliranju: metoda test ne može baciti MyError
C Greška pri kompajliranju: nije moguće naslijediti klasu Error	[✓ **D**] Error caught
E Nema ispisa	**F** Ništa od navedenog

```java
package org.unibl.etf.rt_rk.first;

import java.io.IOException;

public class B {
	public static void main(String args[]) {
		try {
			throw new java.io.IOException();
		}
	}
}
```

11	
A Greška pri kompajliranju: nedostaje finally blok	[✓ B] Greška pri kompajliranju: nedostaje catch blok
C Ispisuje se stack trace koji sadrzi informacije o izuzetku	D Ništa od navedenog

```java
package org.unibl.etf.assignment08;

public class C {
        public static void main(String[] args) {
                Integer i1 = 2;
                Integer i2 = 3;
                Integer i3 = 1;
                Integer i4 = 6;
                Integer i5= i1 & i2 | i3 ^ i4;
                System.out.println(i5);
        }
}
```

^ → XOR

1	0	1
0	1	1
0	0	0
1	1	0

$$\begin{array}{r} 10 \\ 11 \\ \hline \& \quad 10 \end{array}$$

$$\begin{array}{r} 001 \\ 110 \\ \hline \wedge \quad 111 \end{array}$$

$$\begin{array}{r} 10 \\ 111 \\ \hline | \quad 111 \end{array} = 7$$

12	
A 6	B 8
[✓ C] 7	D 1
E 5	F 9

```
/*
 * Maksimalna veličina heap-a je 1500 MB
 * Pretpostaviti da će garbage collector biti pokrenut u trenutku kada na heap-u
nema dovoljno prostora za smještanje novih objekata
 */

package org.unibl.etf.assignment08;

public class D {
	static int counter = 0;
	int intArray[];
	double doubleArray[] = new double[25_000_000];

	public D() {
		intArray = new int[25_000_000];
		byte byteMemory[] = new byte[500_000_000];
		System.out.print(++counter + " ");
	}

	public static void main(String[] args) {
		D array[] = new D[100_000];
		for(int i=0; i<array.length; i++) {
			array[i] = new D();
		}
	}
}
```

~~13~~

A 1 OutOfMemoryError	**B** 1 2 3 OutOfMemoryError
C 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 … StackOverflowError	**D** 1 2 3 4 5 OutOfMemoryError

```java
package org.unibl.etf.assignment06;

public class A {
        public static void main(String[] args) {
                A a = new A();
                try {
                        a.metoda();
                }catch (Exception t) {
                        System.out.println("catch");
                }finally{
                        System.out.println("finally");
                }
                a.metoda2();                                    // A
        }
        void metoda() throws CE1 {
                        throw new CE2("Error 2");
        }
        void metoda2() throws CE3 {
                throw new CE3();                                // B
        }
}
class CE1 extends Exception {
        public CE1() {
                System.out.println("CE1 - 1");
        }
        public CE1(String s) {
                System.out.println(s);
        }
}
class CE2 extends CE1 {
        public CE2(String s) {                                  // C
                System.out.println("CE2 - 2");
        }
}
class CE3 extends RuntimeException {
        public CE3() {
                System.out.println("CE3 - 1");
        }
}
```

14

[✓ A] CE1 - 1 CE2 - 2 catch finally CE3 - 1 Propagiran izuzetak u liniji koda označenoj oznakom A	**B** CE1 - 2 CE2 - 2 catch finally
C Greška pri kompajliranju u liniji koda označenoj oznakom C	**D** Greška pri kompajliranju u liniji koda označenoj oznakom B

jer RunTimeException nije uhvaćen i obrađen nigdje

```java
package org.unibl.etf.assignment09;

public class A {
        public static void main(String[] args) {
                try (AException aEx = new AException()) {
                        System.out.println("try block");
                        aEx.test();
                } catch (Exception e) {
                        System.out.println("catch block");
                } finally {
                        System.out.println("finally block");
                }
        }

}
```

OBAVEZNO

```java
class AException implements AutoCloseable {
        @Override
        public void close() throws Exception {
                System.out.println("close()");
                throw new Exception();
        }
        public void test() {
                System.out.println("test()");
        }
}
```

15	
A Greška pri kompajliranju - catch i finally blokovi su suvišni, jer se koristi try-with-resources konstrukcija	**B** try block test() catch block finally block close()
C try block test() catch block close() finally block	**D** ✓ try block test() close() catch block finally block

```
package org.unibl.etf.assignment10;

interface A {
        void main(String[] args);
}

interface B {
        public void main(String[] args);
}

interface C {
        public static void main(String[] args);
}

interface D {
        protected void main(String[] args);
}

interface E {
        private void main(String[] args);
}
```

* static i private metode u interfejsu moraju imati implementaciju

* protected zabranjen pristup u interfejsima

16	
A Validne su deklaracije interfejsa A, B i C	**B** Validne su deklaracije interfejsa C i E
[✓ **C**] Validne su deklaracije interfejsa A i B	**D** Validne su deklaracije interfejsa A, C i D
E Validne su deklaracije svih interfejsa	**F** Nema validnih deklaracija interfejsa

```
package org.unibl.etf.assignment11;

interface I1 { }

interface I2 { }

class B2 implements I1 { }

class B3 extends B2 implements I2 { }

public class B {
        public static void main(String args[]) {
                B2[] base = { new B2() };                    // A
                B3 dev[] = { new B3() };                     // B
                Object obj = dev;                            // C
                B2 b = obj;                                  // D
        }
}
```

neophodno kastovanje u B2

B2 b = (B2) obj;

17	
A Greška pri kompajliranju u liniji označenoj oznakom A	**B** Greška pri kompajliranju u liniji označenoj oznakom B
C Greška pri kompajliranju u liniji označenoj oznakom C	[✓ **D**] Greška pri kompajliranju u liniji označenoj oznakom D
E Izuzetak pri izvršavanju	**F** Nema izuzetka, nema ispisa

```
package org.unibl.etf.assignment11;

interface I {
        void method();
}

class C implements I {
        static C r = new C();                                  // A
        public static void main(String[] args) {
                C r = new C();                                 // B
                r.method();
        }
        void method() {                                        // C
                System.out.println("abcdef");
        }
}
```

package-private po default-u

OBAVEZNO public

ne smije se smanjiti vidljivost

18	
A Greška pri kompajliranju u liniji označenoj oznakom A	**B** Greška pri kompajliranju u liniji označenoj oznakom B
[✓ **C**] Greška pri kompajliranju u liniji označenoj oznakom C	**D** abcdef

```java
package org.unibl.etf.assignment11;

public enum D1 {
        A (1),
        B (2),
        C (3),
        D (4),
        E (5);
        private D1(int a){
                System.out.print("D1");
        }
        public static void main(String[] args) {
                for(D1 q2: D1.values()){
                        System.out.print(q2);
                }
        }
}
```

.values() vraća niz [A, B, C, D, E]

19	
[✓ A] D1D1D1D1D1ABCDE	B D1D1D1D1D112345
C 12345	D ABCDE
E Izuzetak pri izvršavanju	F Greška pri kompajliranju

```
package org.unibl.etf.assignment11;

public class E {
        private G g = new G();        problematična linija koda
        public E() {
                System.out.println("E");
        }
        void method() {
                System.out.println("method from E");
        }

        public static void main(String[] args) {
                G g = new G();        pokreće rekurzivni lanac
                g.method();
        }
}

class F extends E {
        public F() {
                System.out.println("F");
        }
        void method() {
                System.out.println("method from F");
        }
}

class G extends F {
        public G() {
                System.out.println("G");
        }
        void method() {
                super.method();       poziva metodu SAMO iz neposredne nadklase (u ovom slučaju klase F)      // A
                System.out.println("method from G");
        }
}
```

20	
A E F G method from F method from G	**B** E F G method from E method from F method from G
C E G F G method from F method from G	**D** E F G E F G method from F method from G
[✓ **E**] Izuzetak pri izvršavanju - StackOverflowError	**F** Greška pri kompajliranju u liniji označenoj oznakom A