1. Analizirati kod u sljedećim primjerima i utvrditi da li se može kompajlirati i izvršiti. Ako kod nije moguće kompajlirati ili izvršiti, označiti „problematične" linije koda i navesti razloge. Ako se kod može kompajlirati i izvršiti, napisati izlaze. Po potrebi detaljno obrazložiti.

a) // A1.java

```
public class A1 {
	private A1 a1;
	static{
		System.out.println("A1-S");
	}
	{
		System.out.println("A1-N");
	}
	public A1() {
		System.out.println("A1");
	}
	public A1(A1 a1){
		System.out.println("A1(A1)");
		this.a1 = a1;
		new A2(a1);
	}
	void metoda(){
		System.out.println("metoda A1");
	}
	public static void main(String[] args) {
		A1 a1 = new A1();
		System.out.println("=1=");
		A2 a2 = new A2();
		System.out.println("=2=");
		A3 a3 = new A3(a2, a1);
		System.out.println("=3=");
	}
}

class A2 extends A1 {
	A1 a1 = new A1();
	static{
		System.out.println("A2-S");
	}
	{
		System.out.println("A2-N");
	}
	public A2() {
		this(new A1());
		System.out.println("A2");
	}
	public A2(A1 a1){
		this.a1 = a1;
		System.out.println("A2(A1)");
	}
	private void metoda2(){
		System.out.println("metoda2 A2");
	}
}
```

```
class A3 extends A2 implements Serializable {
      static{
            System.out.println("A3 S");
      }
      {
            System.out.println("A3 N");
      }
      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)");
      }
      void metoda(){
            System.out.println("metoda A3");
      }
      void metoda2(){
            System.out.println("metoda2 A3");
      }
}
```

```
// A1.java

public class A1 {
	private A1 a1;
	{
		System.out.println("A1-N");
	}
	public A1() {
		System.out.println("A1");
	}
	void metoda(){
		System.out.println("metoda A1");
	}
	public static void main(String[] args) {
		A3 a3 = new A3();
		a3.metoda();
		a3.metoda2();
	}
}

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 metoda2(){
		System.out.println("metoda2 A2");
	}
}

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

```java
// T2.java

public class T2 {
	private void m1() {
		System.out.println("1");
	}
	public static void main(String[] args) {
		new TA(){
			void m1() {
				System.out.println("1");
			}
		}.m2();
	}
}

interface TI{
	void m1();
	void m2();
}

abstract class TA implements TI{
	public void m2(){
		System.out.println("2");
	}
}
```

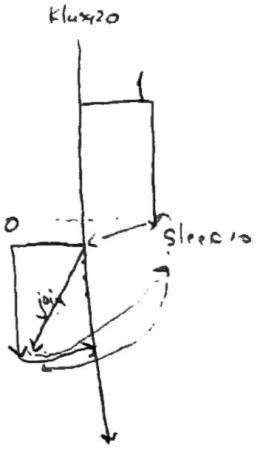
Неименована класа, наследуја TA

```java
// T4.java

public class T4 {
	public static void main(String[] args) {
		Map<Integer, Integer> map = new HashMap<>();
		map.put(10, 10);
		map.put(11, 11);
		map.put(new Integer(10), 100);
		map.put(11, 111);
		map.entrySet().stream().forEach(e -> {
			System.out.println(e.getKey() + ": " + e.getValue());
		});
	}
}

// T5.java

public class T5<X extends Object> {

	private X x;

	public T5(X x) {
		this.x = x;
	}

	private double getDouble() {
		return ((Double) x);
	}

	public static void main(String args[]) {
		Double d = 10d;
		Integer i = d;
		T5<Integer> a = new T5<Integer>(1);
		System.out.print(a.getDouble());
	}
}
```

```
// C1.java

public class C1 {
	public static C1 ref;
	public static void main(String[] args) {
		C1 c1 = new C1();
		C1 c2 = new C2();
		ref = c1;
		try {
			System.out.println(c2.metoda(c2));
			System.out.println(c2.metoda(c1));
		} catch (CE2 e) {
			System.out.println("C1- CE2 catch");
		} catch (CE1 e){
			System.out.println("C1- CE1 catch");
		} catch (Throwable e){
			System.out.println("exception");
		} finally{
			System.out.println("finally");
		}
		System.out.println(c2.metoda(ref));
	}
	int metoda(C1 c) {
		if(c instanceof C1){
			System.out.println("method 1");
		}else
			throw new CE2();
		return 1;
	}
}
class C2 extends C1{
	int a[] = new int[3];
	int metoda(C1 c) throws RuntimeException{
		try{
			if(errorCheck() && c instanceof C2)
				throw new CE2("Error 2");
			else if(errorCheck() && c instanceof C1)
				return a[3];
			else
				throw new CE1("Error 1");
		}catch(CE1 e){
			System.out.println("C2 - CE2");
		}
		ref = null;
		return 0;
	}
	boolean errorCheck(){
		return true;
	}
}
class CE1 extends RuntimeException {
	public CE1(String s) {
		System.out.println("CE1 - 2");
	}
}
class CE2 extends CE1 {
	public CE2() {
		super("s");
	}

	public CE2(String s) {
		super(s);
		System.out.println("CE2 - 2");
	}
}
```

```java
// Klasa20.java

public class Klasa20 extends Thread{
	public Klasa20() {
		System.out.println("Klasa20()");
	}
	public static void main(String x[]) {
		new Klasa20().start();
	}

	public void run() {
		System.out.println("first");
		Thread2 niz[] = {new Thread3(1), new Thread2(), new Thread3(3) };
		for (Thread2 e : niz) {
			if (e instanceof Thread3)
				new Thread(e).start();
			else{
				try {
					e.start();
					e.join();
				} catch (InterruptedException e1) {
					e1.printStackTrace();
				}
			}
		}
		System.out.println("last");
	}
}

class Thread2 extends Thread {
	static int c = 10;
	int id;

	public Thread2() {
		this(0);
	}

	Thread2(int id) {
		System.out.println("Thread2()");
		this.id = (id>0)?id:c++;
	}

	public void run() {
		for (int i = 1; i < 6; i++) {
			try {
				sleep(10);
			} catch (InterruptedException e) {
				e.printStackTrace();
			}
			System.out.println("Thread2 - " + id + ": " + i);
		}
	}
}

class Thread3 extends Thread2 implements Runnable {
	Thread3(int id) {
		super(id);
		System.out.println("Thread3()");
	}
}
```

Klasa20



Klasa20
first
Thread2()
Thred3()
Thread1()
Thread2()
Thread3()
Thread2-1-1
Thread2-0-1

1 - 2
10 - 2
1 - 3
10 - 3
1 - 4
10 - 4
1 - 5
10 - 5
last
3 - 1
3 - 2
3 - 3
3 - 4
3 - 5

2. Za programski kod sa slike napisati izlaz, te kreirati odgovarajuće memorijske reprezentacije koje obuhvataju stanja *stacka*-a i *heap*-a nakon izvršavanja linija koda označenih brojevima 1, 2 i 3. Pretpostaviti sljedeće: da je maksimalna veličina heap-a 1250 MB i da će *garbage collector* biti pokrenut odmah po pozivu `System.gc()` i u trenutku kada je na *heap*-u nema dovoljno prostora za smještanje novih objekata. (6)

```
// M1.java

public class M1 {
	int id;
	M1 m;
	M2 m2;
	double dm[] = new double[2_500_000];
	long lm[] = new long[2_500_000];
	int im[] = new int[2_500_000];

	public M1(M1 m, int id){
		System.out.println("M1 " + id);
		this.m = m;
		this.id = id;
		m2 = new M2();
	}
	@Override
	protected void finalize(){
		System.out.println("M1 finalize");
	}
	public static void main(String args[]){
		M1 m1 = new M1(null, 1);
		M1 m2 = new M1(m1, 2);
		M1 array[][] = new M1[2][3];
		for (int i = 0; i < array.length; i++) {
			for(int j = 0; j < array[i].length; j++) {
				array[i][j] = new M1(m1, 0);
			}
		}
		System.gc();                                    // 1
		array[0] = null;
		System.gc();                                    // 2
		M1 arr[] = {new M1(m1, 9), new M1(m1, 10)};
		array[0] = arr;                                 // 3
		array[0][2].m2 = new M2();
	}
}

class M2{
	float fm[] = new float[12_500_000];             // 100 MB
	public M2() {
		System.out.println("M2");
	}
	@Override
	protected void finalize(){
		System.out.println("M2 finalize");
	}
}
```

d) // A1.java

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

class A2 extends A1 {
	A1 a1 = new A1();
	static{
		System.out.println("A2-S");
	}
	{
		System.out.println("A2-N");
	}
	public A2() {
		this(new A1());
		System.out.println("A2");
	}
	public A2(A1 a1){
		this.a1 = a1;
		System.out.println("A2(A1)");
	}
	private void metoda2(){
		System.out.println("metoda2 A2");
	}
}
```

```
class A3 extends A2 implements Serializable {
	static{
		System.out.println("A3-S");
	}
	{
		System.out.println("A3-N");
	}
	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)");
	}
	void metoda(){
		System.out.println("metoda A3");
	}
	void metoda2(){
		System.out.println("metoda2 A3");
	}
}

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

```java
// Klasa3.java

public class Klasa3{
	static int x = 3;
	public static void main(String[] args) {
		new Klasa3();
	}
	Klasa3() {
		Klasa3(2);
	}
	Klasa3(int x) {
		System.out.println(x);
	}
}
```

```java
// Klasa21.java

class A1 {
	private A1 a1;
	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");
	}
}
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)");
	}
	private void metoda2(){
		System.out.println("metoda2 A2");
	}
}

class A3 {
	public A3() {
		System.out.println("A3");
	}
}

class Klasa21 extends A3 {
	private A1 a = new A2();
	private A2 a2 = new A2(new A1());
	public Klasa21() {
		a2.metoda();
		System.out.println("A4");
		a.metoda();
	}
	public static void main(String[] args) {
		Klasa21 a4 = new Klasa21();
		a4.metoda();
	}
	protected void metoda(){
		System.out.println("metoda Klasa21");
	}
}
```

A3
A1
A4
A2(A1)
A2
A1
A1
A2(A1)
metoda A1
A4
metoda 21
metoda Klasa21

```java
// Klasa7.java

public class Klasa7 {
	public Klasa7() {
		System.out.println("Klasa 7");
	}
	public static void main(String[] args) {
		Klasa7 e = new Klasa7();
		Klasa8 f = new Klasa8();
		try {
			f.metoda();
			e.metoda();
		}catch (Exception t) {
			System.out.println("catch 1");
		}finally{
			System.out.println("finally");
		}
		e.metoda2();
	}
	void metoda() throws CE1 {
			throw new CE2("Error 2");
	}
	void metoda2() throws CE3 {
		throw new CE3();
	}
}
class Klasa8 extends Klasa7{
	public Klasa8() {
		System.out.println("Klasa 8");
	}
	void metoda(){
		try{
			throw new CE1();
		}catch (CE1 e) {
			System.out.println("catch 2");
		}
	}

}
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() {
		System.out.println("CE2 - 1");
	}
	public CE2(String s) {
		System.out.println("CE2 - 2");
	}
}
class CE3 extends RuntimeException {
	public CE3() {
		System.out.println("CE3 - 1");
	}
}
```

Klasa 7
Klasa 8
CE1 - 1
catch 2
CE1 - 1
CE2 - 2
catch 1
finally
CE3 - 1
Exception

```java
// Klasa0.java

public class Klasa0 {
	static {
		int x = 5;
	}

	static int x, y;

	public static void main(String args[]) {
		x--;
		System.out.println(x + " " + y);
		metoda();
		System.out.println(x + " " + y);
		System.out.println(++x + x++);
		System.out.println(++Klasa0.x);
	}

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

// Klasa1.java

public class Klasa1 {
	int i = 0;

	public static void main(String argv[]) {
	}

	Klasa1() {
		top: while (i < 2) {
			System.out.println(i);
			i++;
			continue top;
		}
	}
}


// Klasa2.java

class Klasa2{

	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);
		metoda2();
		System.out.println(i + y);
		System.out.println(x + j);
	}

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

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

```java
// Klasa4.java

public class Klasa4 {
	int j = 1;

	public static void main(String args[]){
		metoda();
		Klasa4 a = new Klasa4();
		a.metoda();
	}
	public static void metoda(){
		char digit = 'a';
		for (int i = 0; i < 10; i++){
			switch (digit){
				case 'x':
				{
					int j = 0;
					System.out.println(j);
				}
				default:
				{
					int j = 100;
					System.out.println(j);
				}
			}
		}
		int i = j;
		System.out.println(i);
	}
}
```

```java
// Klasa5.java

public class Klasa5 {
	int x = 0, y = 0;
	Klasa5(int a, int b){
		x = a;
		y = b;
	}
	protected int zbir(){
		return x + y;
	}
	protected int razlika(){
		return x - y;
	}
	public static void main(String s[]){
		Klasa5 b = new Klasa5(1,2);
		Klasa6 c = new Klasa6();
		System.out.println(b.razlika());
		System.out.println(c.razlika());
	}
}

class Klasa6 extends Klasa5{
	public int zbir(){
		return y+x;
	}
	public int razlika(){
		return y-x;
	}
}
```