```
public class Bariera {

	//pocet vlaken narazejici do bariery (viz v sesite obr)
	private int bariera = 0;

	//nascitavani pro vypocet pi
	private double sum = 0;

	//jaka bude vzdalenost mezi jednotlivymi barierami
	private int synchronizace;



	Bariera(int synchronizace){
		this.synchronizace = synchronizace;
	}



	synchronized public void pricti(Vlakno v) throws InterruptedException{


		if(v.getAktualniClen() % synchronizace == 0){
			//vlakno je nutno pricist
			bariera++;


			if(bariera == v.getPocetVlaken()){
				sum += v.getSoucet();

				System.out.println("Cislo pi po souctu"
				+ v.getAktualniClen() * v.getPocetVlaken()
				+ ", clenu " + 4* sum);

				bariera = 0;


				//probudi vsechny vlakna nad timto objektem
				notifyAll();

				System.out.println("Synchronizace dokoncena, probouzim
				                    ostatni vlakna");
			}else{
				sum += v.getSoucet();
				try{


				//bude uspano vlakno
				wait();

				} catch(InterruptedException e){
					e.printStackTrace();
				}
			}
		}
	}
}
```

```java
public class Vlakna {

	public static void main(String[] args) throws InterruptedException {

		// TODO Auto-generated method stub
		final int pocet_clenu = 100;

		final int pocet_vlaken = 5;

		final int synchronizace = 10;

		int i;

		double suma = 0;


		Bariera bar = new Bariera(synchronizace);


		Vlakno v[] = new Vlakno[pocet_vlaken];


		for (i = 0; i < pocet_vlaken; i++) {


			v[i] = new Vlakno(i, pocet_clenu, pocet_vlaken,bar);


			v[i].start();

			suma += v[i].getSoucet();
		}


		for(i = 0; i<pocet_vlaken;i++){
			try{


			v[i].join();

			}catch(InterruptedException e){
				e.printStackTrace();
			}


			suma += v[i].getSoucet();
		}
		System.out.println("Vysledne cislo pi: " + 4 * suma);
	}
}
```

```
class Vlakno extends Thread {

	private int cislo_vlakna;

	private int pocet_clenu;

	private double soucet;

	private int aktualni_clen = 0;

	private int pocet_vlaken;


	private Bariera bar;



	Vlakno(int cislo_vlakna, int pocet_clenu, int pocet_vlaken, Bariera bar) {
		this.cislo_vlakna = cislo_vlakna;
		this.pocet_clenu = pocet_clenu;
		this.pocet_vlaken = pocet_vlaken;
		this.bar = bar;
	}


	public double getSoucet() {
		return this.soucet;
	}


	public int getAktualniClen(){
		return this.aktualni_clen;
	}


	public int getPocetVlaken(){
		return this.pocet_vlaken;
	}



   public void run() {

		int index;

		//vypocet hodnoty prvniho prvku
		soucet = Math.pow(-1, cislo_vlakna) / (2 * cislo_vlakna + 1);


		aktualni_clen++;



		while (aktualni_clen <= pocet_clenu) {

//kriticka sekce - pridanim  klicoveho slova synchronized se prelozeni povede
			try {
```

```
                        bar.pricti(this);


                } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }

                index = cislo_vlakna + pocet_vlaken * aktualni_clen;



                soucet += Math.pow(-1, index) / (2 * cislo_vlakna + 1 + 2 *
                pocet_vlaken * aktualni_clen);

                aktualni_clen++;
        }

        System.out.println("Vlakno "+cislo_vlakna+" ma soucet: "+ soucet);
    }
}
```