Polimorfizmus mítoszok

Az objektum orientált programozás terjedésével a monolitikus programokban *hívk* egyik nagy ellenérve az öröklődéssel és a polimorfizmussal kapcsolatban a túlzott erforrásigény volt. A JavaSpecialist blog gazdája készített egy mérést, amely alapján mi is készítettünk méréseket, amelyek meglep eredménnyel zárultak. Egészséges Java öntudattal az ember azt kell gondolja, hogy az *interface* vagy az *abstract* osztályok használata nem lassít a program futásán, kis gyanakvással pedig úgy gondolja, hogy *"jó-jó, biztos lassít valamit, de nem sokat"*. Nosza, mérésre fel. Az alapkoncepció egyszeren az, hogy írjunk egy csomó osztályt, amelyek megvalósítják a tesztelend folyamatokat:

Used.java

```
public class Used
{
  public int echo(int param)
  {
    return param;
  }
}
```

illetve

Use.java

```
public class Use
{
  private final Used used;
  public Use(Used used)
  {
    this.used = used;
  }

  public int run(int i)
  {
    return used.echo(i);
  }
}
```

Ezekbl kiindulva csinálunk sima *interfész-implementációt*, *többszörös implementációt*, *absztrakt-kiterjesztést*, *többszörös absztakt-kiterjesztést*, illetve *absztrakt-kiterjesztést* továbbörökítünk. A program megméri, hogy 500 millió ciklus mennyi id alatt fut le a *run* metódusból, illetve minimumot, maximumot és átlagot számol. Nézzünk néhány mérési eredményt, amelyek különféle JVM-en készültek:

	1.6.0_03-b05	1.5.0_14-b03	1.6.0_01-b06	1.5.0_07-b03
Inline	1067ms	575ms	896ms	895ms
Simple	1921ms	1144ms	1833ms	1641ms
One implement	1931ms	1448ms	2313ms	1858ms
Two implement	1946ms	1206ms	2314ms	1605ms
One abstract	1961ms	1228ms	1834ms	1604ms
Two abstract	1147ms	1224ms	1835ms	1604ms
Override	1482ms	1213ms	2310ms	1856ms

Mint látható, az eredmények igen változatosak... elgondolkodtató, hogy melyik eredményt miért kapjuk... :)

A (NetBeans) projekt, forrásokkal együtt letölthet a Polymorph.zip.