Virtual machines in JR

virtual machine created by

```
vm c1 = new vm();
vm c2 = new vm() on "opsys13.tic.heia-fr.ch" // string or vm reference
```

creating remote objects

```
remote Foo f = new remote Foo(11); // any class from which remote objects are
                                   // instantiated must be public
- or -
remote Foo f = new remote Foo(11) on c1;
```

Can't access static members of a remote object reference.
Be careful about parameter passing.

Virtual Environment

Virtual machines

You have the following VMs that you can use:

- `opsys11.tic.heia-fr.ch`
- `opsys12.tic.heia-fr.ch`
- `opsys13.tic.heia-fr.ch`
- `opsys14.tic.heia-fr.ch`
- `opsys15.tic.heia-fr.ch`
- `opsys16.tic.heia-fr.ch`
- `opsys17.tic.heia-fr.ch`
- `opsys18.tic.heia-fr.ch`

They all share the same userhome directory --> login on each of these 8 machines is the same and also the same files appear.

Access

You can access with the same access that you got for your CVS access.
!! The password is the one that you got at the beginning of the semester !!

Environment Variables

Make sure that environment variables are correctly set in $HOME/.bashrc:

```
export DISPLAY=":0.0"
export JRSH="/bin/bash"
export JR_HOME="/common/jr"
export JRRSH="/usr/bin/ssh"
export CLASSPATH=".:$JR_HOME/classes/jrt.jar:$JR_HOME/classes/jrx.jar:$CLASSPATH"
export JAVA_HOME="/common/jdk"
export PATH="$JR_HOME/bin:$JAVA_HOME/bin:$JAVA_HOME/jre/bin:$PATH"
```

Setting up ssh to allow passwordless access

Normally this shouldn't be necessary.... already done

`ssh-keygen` *(answer yes to all questions)*

`cd ~/.ssh`

`cp id_rsa.pub authorized_keys`

and that's it. They then need to `ssh` on each remote machine they want to use at least once to answer `yes` to the question, and the question should no longer be asked.

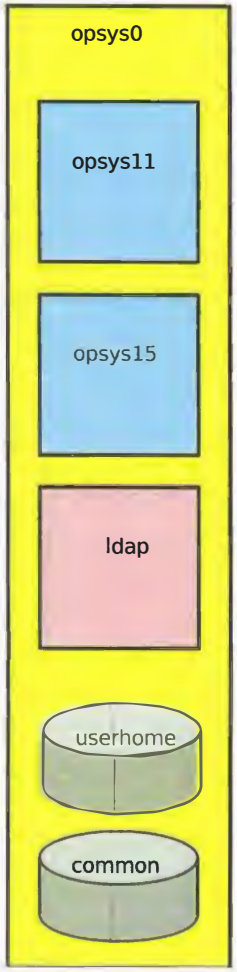
```
The authenticity of host 'opsys16.tic.heia.fr.ch (160.98.22.60)' can't be established.
ECDSA key fingerprint is SHA256:p9r4w+6Sfht614FxrKb5Ac305g3PojzcCVyNCgIC3bM.
Are you sure you want to continue connecting (yes/no)?
```

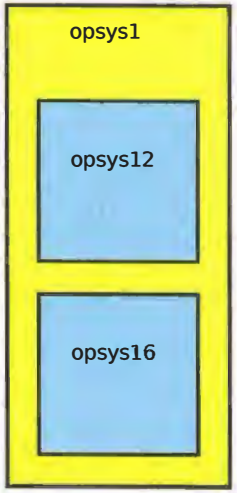
So, for example,

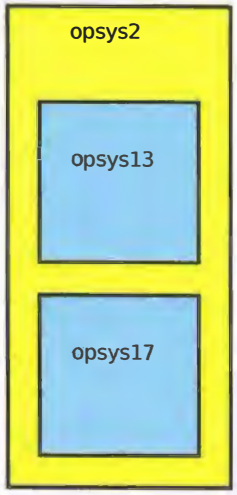
`ssh opsys11.tic.heia-fr.ch date`

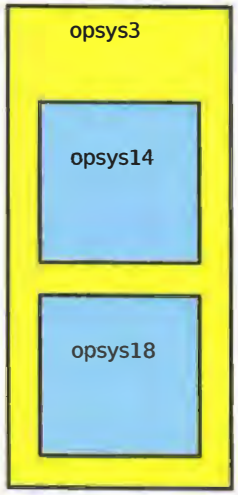
and answer `yes` to the question. Should get data/time from remote system. Next time they `ssh opsys11.tic.heia-fr.ch` the question should not be asked. See the jr web site, https://web.cs.ucdavis.edu/~olsson/research/jr/versions/2.00608/instructions.html where there is a discussion of what to check to get multiple VMs running.

Infrastructure pour programmation distribuée et programmation concurrente









ssh <username>@opsys1<x>.tic.heia-fr.ch avec <x> de 1 à 8

java et jr sont dans /common

IP : 160.98.22.55 (opsys11) à 160.98.22.62 (opsys18)

```
import edu.ucdavis.jr.JR;

// a simple class to place remotely
public class R {

   // get two ints from master
   public op void f(int []);

   // used to make output deterministic
   public op void givesem(cap void ());

   // receive integers from 2 different sends
   process p {
      int a[];  // get some integers from master
      System.out.println("in p");
      cap void () go;  // semaphore to make output deterministic

      receive givesem(go);  // all messages have been sent
      for (int k = 1; k <= 2; k++) {
         receive f(a);
         for (int i = 0; i < 2; i++) {
            System.out.println(k +" a["+i+"] "+a[i]);
         }
         V(go); // tell master I'm done
      }
   }
}
```

```
import edu.ucdavis.jr.JR;

// show what happens on different VMs
// Other than placement on different vm, this code is
// identical to SameVMArraySend
public class DiffVMArraySend {

   // no arguments required
   public static void main(String [] args) {
      remote R rr = new remote R() on new vm(); //same phys mach
      int [] b = new int [2];      // any type would do
      b[0] = 11; b[1] = 34;
      sem go;                     // to make output deterministic
      send rr.f(b);
      b[0] = 65; b[1] = 87;
      send rr.f(b);
      send rr.givesem(go); // I'm done, wait for child to finish
      P(go);
   }
}
```

```
import edu.ucdavis.jr.JR;

// a simple class to place remotely
public class R {

   // get two ints from master
   public op void f(int []);

   // used to make output deterministic
   public op void givesem(cap void ());

   // receive integers from 2 different sends
   process p {
      int a[];  // get some integers from master
      System.out.println("in p");
      cap void () go;  // semaphore to make output deterministic

      receive givesem(go);  // all messages have been sent
      for (int k = 1; k <= 2; k++) {
         receive f(a);
         for (int i = 0; i < 2; i++) {
            System.out.println(k +" a["+i+"] "+a[i]);
         }
         V(go); // tell master I'm done
      }
   }
}
```

```
import edu.ucdavis.jr.JR;

// show what happens on different VMs
// Other than placement on different vm, this code is
// identical to DiffVMArraySend
public class SameVMArraySend {

   // no arguments required
   public static void main(String [] args) {
      remote R rr = new remote R(); //same virt mach
      int [] b = new int [2];      // any type would do
      b[0] = 11; b[1] = 34;
      sem go;                      // to make output deterministic
      send rr.f(b);
      b[0] = 65; b[1] = 87;
      send rr.f(b);
      send rr.givesem(go); // I'm done, wait for child to finish
      P(go);
   }
}
```