# PXC Pràctica 3 - RMI

Pere Joan Martorell Calaf
Bernat Farrero Badal

RMI és un mecanisme que Java ofereix per invocar un objecte de manera remota. Serveix per crear aplicacions distribuides en entorns íntegrament de Java.

**Per executar la pràctica** (un cop descomprimida):

1. Compilar:

    *javac *.java*

2. Crear els Stub i Skeletons (representant en client i representat en servidor) de l'aplicació amb la comanda *rmic*:

    *rmic lloguerDeCotxesImpl* (al servidor i al client)

3. Posar en marxa el registre amb la comanda *rmiregistry.* El registre s'encarrega de buscar i registrar objectes remots.

4. Executar servidor i client (teòricament en dos màquines diferents, però si és en una en dos consoles o afegir &).

    *java lloguerDeCotxesServidor &*
    *java lloguerDeCotxesClient*

Nota: El port 1099 ha d'estar lliure, ja que és el port que usem per defecte.

**Com implementar les classes**

Per escriure el codi hem partit de l'exemple i les capçaleres que ens venien donades en l'exemple de l'enunciat de la pràctica. Hem decidit usar la classe InOut per escriure i llegir per pantalla. Aquesta vegada l'interfície de comunicació amb l'usuari no serà un navegador sinó la pròpia terminal.

La classe que captura els resultats de la interacció amb el client és lloguerDeCotxesCliente.java. El que fa és oferir per pantalla les mateixes opcions de lloguer de les anteriors pràctiques i recollir els resultats introduits pel client. Això es fa per mitjà d'una funció creada que mostra el menú.

En el cas de crear un lloguer, i un cop les dades introduides han estat degudament validades (com en les anteriors pràctiques), es procedeix a la crida de la funció lloga(). En el cas que es demani la segona opció de llistar lloguers, es procedeix a cridar llistaLloguers(). Ambdues funcions contenen una part clau que val la pena mencionar explícitament, la cerca de l'objecte distribuit al servidor:

```
lloguerDeCotxes c = (lloguerDeCotxes)Naming.lookup("rmi://localhost:1099/ServiciolloguerDeCotxes");

c.reglloguer(model, submodel, dies, numv, desc, total);

lloguerDeCotxes c = (lloguerDeCotxes) Naming.lookup ("rmi://localhost:1099/ServiciolloguerDeCotxes");

c.llistarLloguers(user,pw);
```

Aquests objectes són declarats i definits al servidor, en particular a la classe d'implementació *lloguerDeCotxesImpl*.

En la funció regLloguer(...) cridada pel client, es crea un nou lloguer i s'afegeix a un objecte (definit com a variable global) de tipus llista (ArrayList de Java) format per tuples de lloguers. El lloguer és una altra classe específica definida per nosaltres i que conté els camps d'un lloguer. Com es pot comprovar, gestionem la persistència mitjançant variables globals en comptes de fitxers plans o bases de dades posat que es tracta d'una aplicació senzilla de proves.

En la funció llistarLloguers(...), un cop comprovat que el nom d'usuari i password passats com a paràmetre són correctes (per la nostra pràctica *pxc* i *rmi1234* respectivament) definim un iterador que iteri entre tots els lloguers emmagatzemats i els escrigui per pantalla.

La classe lloguerDeCotxesServidor declara al registre la classe definida anteriorment amb la implementació dels dos mètodes.

Finalment, la classe lloguerDeCotxes conté les dues capçaleres de les crides remotes registrades, tal com indica l'enunciat de la pràctica.