1.Memorie

* REGISTERS(in processor),cel mai rapid mod de stocare,nu ai control direct acolo.(in c,c++ poti sugera alocarea pe registry)
* Stack(RAM),are support de la processor printr-un stack pointer.Stack pointerul se misca in jos pt alocare de mem,in sus pt dezalocare.Java tb sa stie cand incepe programul , lifetime ul pt fiecare obiect pe care il pune in stack.NU AICI SE PUN OBIECTELE,AICI AI DOAR REFERINTE.
* Heap(RAM) aici ai toate obiectele
* Constant storage-hardcodate in cod sau ROM(read-only memory)

Non-ram storage pentru datele care sunt in afara codului si traiesc si cand programul nu ruleaza.ex.streamed obj,persistent obj.Java rezolva asta,dar

LightweightmJDBC si hibernate o fac cum trebuie.

2.Types

* Wrapperul pt clasele primitive te lasa sa faci obiecte non-primitive in heap pt a reprezenta tipul primitiv(Java SE5 face singur in spate treaba asta)
* BigInteger si BigDecimal sunt wrappere,desi nu exista data primitiva.Exista metode pe care le faci ca pt inturi si floaturi ,DAR MULT MAI INCETE.Deci dai viteza pe acuratete(poti sa pui oricate cifre vrei,nu exista limita de size)

3.Scope

* Ca in C si C++,scope-ul e determinat de {}
* La obiecte:exista obiectul cat timp ai o referinta catre el,daca nu ai ref. inseamna ca nu ai cum sa accesezi,deci garbage collector-ul iti sterge obiectul din heap

4.Classes

* Ca idee principala,trebuie initializat absolut tot(camuri,variabile,instante,obiecte)
* Tot ce inseamna metoda de returnare🡪la fel ca c++
* Static:poti apela metoda fara sa faci o instanta a obiectului si se storeaza campul o singura data in memorie ,nu conteaza cate obiecte de tipul respectiv ai
* Promotion:pt char,byte,short ca sa folosesti operatori ,ei sunt adusi (“promovati”) la nivel de int