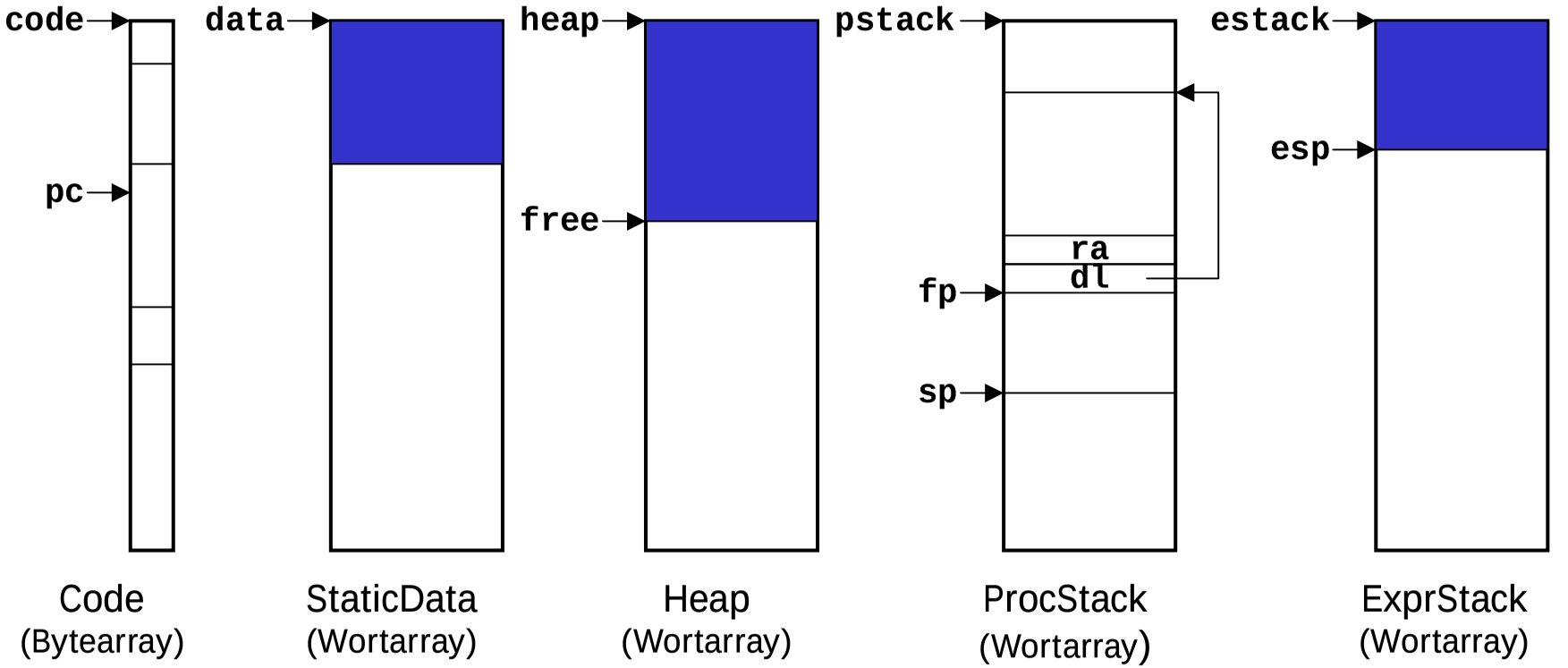
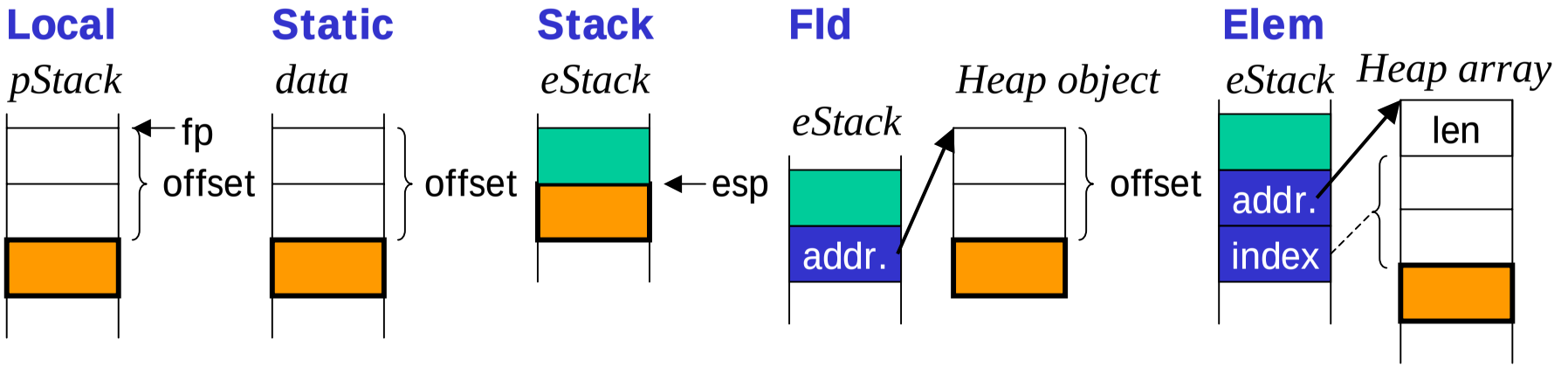
MicroJava VM: Speicher-Layout



Codeitem-Klasse *Item*

```
class Item {
    public static final int   // Item Art
        Con=0, Local=1, Static=2, Stack=3, Fld=4, Elem=5, Meth=6;
    public int kind;
    public Struct type;     // Typ des Operanden
    public Obj obj;         // Meth: Methodenobjekt aus Symbolliste
    public int adr;         // Con: Wert; Local, Static, Fld, Meth: Adresse
}
```



Beispiel: Attributieren einer Grammatik

```
Expr = Term { "+" Term } .
Term = Factor { "*" Factor } .
Factor = number | ident [ "." ident | "[" Expr "]" ].
```

Beispiel: Ablauf für **b[5] + a * 10**

Deklaration:

class X int a; { void m() char c; int b[]; { ... } }

Klasse *Struct* – Hilfsmethoden


```
public boolean isRefType () { return kind == Class || kind == Arr; }
public boolean equals (Struct other) {
    if (kind == Arr) return other.kind == Arr &&
                                elemType.equals(other.elemType);
    if (kind == Class) return other.kind == Class && n == other.n &&
                                  Obj.equalsCompleteList(fields, other.fields);
    return this == other;     // must be same type Obj node
}
public boolean compatibleWith (Struct other) {
    return this.equals(other) ||
           this == Tab.nullType && other.isRefType() ||
           other == Tab.nullType && this.isRefType();
}
public boolean assignableTo (Struct dest) {
    return this.equals(dest) ||
           this == Tab.nullType && dest.isRefType() ||
           this.kind == Arr && dest.kind == Arr &&
           dest.elemType == Tab.noType;     // für len-Funktion
}
```