

```

repeat:
  if (!index)
    goto check;

  sat = PRK_OBJECT_SAT(TABLE_ELT(parent_table, --index));
  l = PRK_SAT_MULTI_SATS(sat, PRK_SAT_HASH_TABLE_SIZE(sat));

  if (Null(l))
    goto create_new;

  if ((u_long)PRK_FIRST(l) >= (u_long)min_multi_sat_count)
    goto repeat;

  min_multi_sat_list = l;
  goto new_min_loop;

check:
  for (min_multi_sat_list = PRK_REST(min_multi_sat_list);
       !Null(min_multi_sat_list);
       min_multi_sat_list = PRK_REST(min_multi_sat_list)) {
    index = 0;
    sat = (PrkSat)PRK_FIRST(min_multi_sat_list);
    l = PRK_SAT_PARENTS(sat);
    for (; !Null(l); l = PRK_REST(l)) {
      if ((PrkObject)PRK_FIRST(l) != TABLE_ELT(parent_table, index))
        break;
      if (index++ == total)
        if (Null(PRK_REST(l)))
          return (PrkSat)PRK_FIRST(min_multi_sat_list);
      else
        break;
    }
  }

create_new:
  return prk_create_new_multi_sat(parent_table);
}

```

40

What is claimed is:

1. In an object-oriented software system on a computer having a central processing unit and a memory means, said object-oriented system having data structures comprising objects, said objects representing real-world entities, said objects being an aggregation of attributes, wherein form of aggregation of said attributes is specified by attribute declarations, a method for generating a data representation of said objects so that source code is independent of value access and for accessing said data representation for use in implementing said objects of the software system, said method comprising the steps of:

using said central processing unit to create an attribute value array for each one of said objects for storing in said memory means specific values of said attributes;

using said central processing unit to store in said memory means said attribute declarations in an attribute access descriptor data structure having an attribute selector and an attribute offset, said attribute selector being the name of said attribute, said attribute offset being an index into said attribute value array;

using said central processing unit to create an attribute access table for each said object and to store all attribute access tables in said memory means; thereafter

using said central processing unit to place in said attribute access tables pointers to said attribute access descriptor data structures for each one of said objects, thereby to provide the data representation of said objects; and thereafter

using said central processing unit to access said specific values of attributes in said attribute value array by specifying said attribute selector, using said pointers to find the attribute access descriptor data structure having said attribute selector, and applying said attribute offset as an index into said attribute value array, thereby to access the data representation of said objects.

2. The method of claim 1, wherein said attribute declarations include an element specifying access type, and wherein said access type element has the value "dynamic access" so as to permit generic access to said attribute values, further comprising the step of;

thereafter using the central processing unit to replace said "dynamic access" value of said access type element of said attribute declarations in said attribute access descriptor data structure with a value "static access" in order to permit direct reference to said specific values in said attribute value array; said attribute declarations specifying layout of internal data structure.

* * * * *