Polymorphism- the objects in a moving train

R K Joshi
IIT Bombay

Polymorphism Demo:
Abstraction hierarchy
lets us write
generic code

Polymorphism Demo:
Abstraction hierarchy
lets us write
generic code

What we have seen so far

Generic Code can be written in terms of the common type

That code becomes applicable to all objects of the specific derived types

Let's work out an example..

```
class Shape {

    int bx,by,bw,bh;

            // outer board limits

  protected:

    int x,y,w,h;

            // bounding box of shape
....
```

The following code is generic

```
bool moveall2left(int k) {

        // by k units

        for (int i=0; i<parts.size(); i++)
                if (!parts[i]->leftok(k)) return false;

        for (int i=0; i<parts.size(); i++)
                parts[i]->moveleft(k);

        return true;
}
```

Replaceability

Generic *g;

Specific1 *s1;

Specific2 *s1;

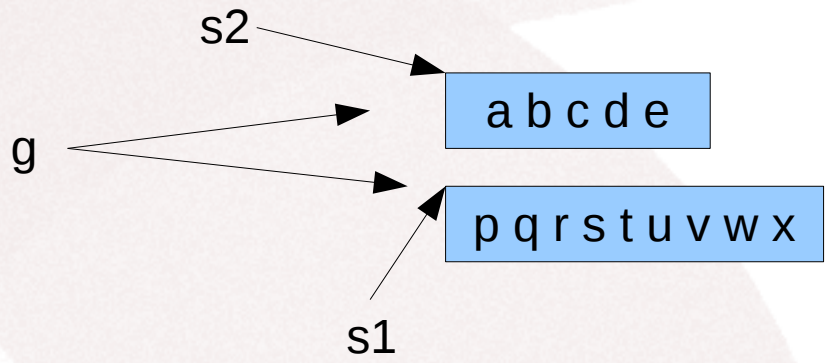
s1 can be assigned to g

s2 can be assigned to g

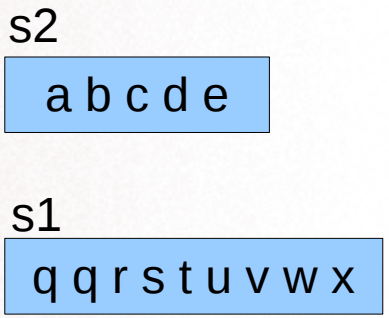
virtual member functions invoked on g result in call to their latest implementations

Polymorphic variables

in c++: those which are pointers



not those which are created statically



More invocations on generic pointer variable of generic type

```
int Board::handle(int e) {


    if (e==FL_KEYDOWN) {

      for (int i=0; i<parts.size();i++)  parts[i]->hide();
      for (int i=0; i<parts.size();i++)  parts[i]->render();
      switch(Fl::event_key()) {
          .....
       }
      ....
    }
...
```

Actions on Board calling Board's member functions in keyboard event handler.. ... these calls contain generic calls

```
case FL_Escape: exit (1);  // Esc key

case FL_Left: movealllleft(10); break;  // left arrow

case FL_Up: moveallup(10); break;  // up arrow

case FL_Right: moveallright(10); break; // right arrow

case FL_Down: movealldown(10); break; // down arr.
```

Creating many specific objects.. These calls are from main, on board, the container

```
for (int i=0;i<4;i++) {
 // creating the train by adding components into board
    int x=100+i*120;
    b->createBlock(x,100,100,50,44+i*20);
    b->createWheel(x+10,150,30,30,58+i*4);
    b->createWheel(x+60,150,30,30,64+i*5);
    b->createBlock(x+100,120,20,10,195);
}
b->createEngineFront(80,100,40,50, 160);
    // this is the final component added
```

Board holds the generic variable

```
class Board : public Fl_Widget {

vector <Shape *> parts;
int color;
int bx, by, bw, bh;
Fl_Box *infobox ;
```