POLYMORPHISM

```
Object   obj = new GeoCountDownTimer(2022, 12, 31)
         obj = new Integer(3);
```

POLYMORPHISM

- "having many forms"
- allows class instance to behave like another ancestor class
- **polymorphic reference** - reference variable that can refer to different objects at different points in time

POLYMORPHISM - METHOD BINDING

- Specific method invoked can vary from one invocation to the next
- Depends on what instance the reference currently points to
- for polymorphic reference it's **dynamic binding** - made at runtime

```
Employee e1 = new StudentWorker();
Employee e2 = new Faculty ();
  e1 = e2;
```

EXAMPLES

Consider classes:

- `Employee` - abstract
- `StudentWorker` - extends `Employee`
- `Faculty` - extends `Employee` Can have
- `Employee` reference point to StudentWorker object
- `Employee` reference point to Faculty object

EXAMPLES - CODE

```
Employee e1;
Employee e2;
StudentWorker s1 = new StudentWorker();
Faculty f1 = new Faculty();
e1 = new StudentWorker();
f1 = new Faculty();
```

POLYMORPHISM - INTERFACE

- Can also do with interface
  - make object reference variable with interface name
  - can refer to any object of classes implementing the interface
- Example:
  - `List` is built-in interface in Java
  - `ArrayList` implements list

```
List lst = new ArrayList();
```

ArrayList lst = new LinkedList();

POLYMORPHIC REFERENCES

- Often used as formal parameter to a method
  - `button.addActionListener(ActionLis`
    `l)`
  - `Collections.sort(List list)`