ITEM 15 :   Date:..............................

Provide access to raw resources in resource managing classes.

→ Many API'S refer to resources directly

→ Item 13 introduces idea of smart ptr to hold the result of a call to a factory functn like CreateInvestment.

→ Suppose we need a function to use working with Investment object

```
int daysHeld (const Investment * pi);
 //returns no. of days Investment has been
   held.
```

But we would like to call it like this

```
int days = dayHeld(pInv) ; //error.
```

daysHeld want raw Investment * ptr & we are passing object of type shared.
↳ std:: tr1::shared <Investment> pInv (CreateInvest

```
int days = dayHeld (pInv.get());
```

//fine, passes the raw ptr in pInv to daysHeld.

//above one is explicit conversion.

Date:......................................

Implicit conversion.

```
class Investment {
    public:
        bool isTaxFree() const;
};
Investment * createInvestment();

std::tr1::shared <Investment> pi1 (createInvst());

bool taxable1 = !(pi1 -> isTaxFree());

bool taxable2 = !((*pi2).isTaxFree());
```

// we have overloaded the deferencing ~~ptrs~~ operators (operato -> & operator(*));

taxable1 access resource via operator ->
taxable2 access resource via operator *

Date:....................................

Often explicit conversion function like get is the prefeeable path. becoz it minises the chaces of unintended type conveesions.

```
~~int~~ double a = 10.5;
        int i;

        i = a; // implicit conversion

      i = (int) a; //explicit conversion.
```

explicit is better since it will ~~at~~ not lead to any confusion & will be clear to compiler of what programmer wants.