Child c = new Child();
Parent p = getSomeRandomCrazyObject();

c = p;

p = c;
Child c = new Child();
Parent p = getSomeRandomCrazyObject();

// the following results in a compile time (javac) error
// b/c Parent objects in general do not have Child parts
// which is assumed for any Child type \textit{variable}

\begin{quote}
\texttt{c = p; error: incompatible types: Parent cannot be converted to Child}
\end{quote}

// OK (no error) b/c c must have "Parent parts"

p = c;
Parent p = getObject();

val = p.field1;
//what is in val?

class Parent {
    ...
    int field1 = 10;
    int field2 = 20;
}
class Child
extends Parent {
    ...
    int field1 = 5;
    int field3 = 4;
}
Parent p = getObject();

val = p.field1;
//what is in val? 10

//what if getObject returns a Child type object?

class Parent {
    ...
    int field1 = 10;
    int field2 = 20;
}
class Child extends Parent {
    ...
    int field1 = 5;
    int field3 = 4;
}
Parent p = getObject();

val = p.field1;
//what is in val? 10

//what if getObject returns a Child type object? 10

//what happens here?
p = new Parent();
val = p.field3;

class Parent {
    ...
    int field1 = 10;
    int field2 = 20;
}
class Child
    extends Parent {
        ...
        int field1 = 5;
        int field3 = 4;
    }

Parent p = getObject();

val = p.field1;
//what is in val?  10
//what if getObject returns a Child type object?  10

//what happens here?
p = new Parent();
val = p.field3;  //ERROR, no field3 in type Parent or  
//supertype of Parent

class Parent {
    ...
    int field1 = 10;
    int field2 = 20;
}
class Child
    extends Parent {
    ...
    int field1 = 5;
    int field3 = 4;
}
//and here?
Child c = new Child();
val = c.field2;

class Parent {
    ...
    int field1 = 10;
    int field2 = 20;
}
class Child
    extends Parent {
        ...
        int field1 = 5;
        int field3 = 4;
    }
//and here?
Child c = new Child();
val = c.field2;  // 20

//how do I get to Parent's field1 from the child?
//and here?
Child c = new Child();
val = c.field2;

//how do I get to Parent's field2 from the child?
val = ((Parent)c).field1;

//how do I access f1 and f2?

```java
class Parent {
    ...
    int field1 = 10;
    int field2 = 20;
    static int f1 = 4;
    static int f2 = 7;
}

class Child extends Parent {
    ...
    int field1 = 5;
    int field3 = 4;
    static int f1 = 2;
}
```
//and here?
Child c = new Child();
val = c.field2;

//how do I get to Parent's field2 from the child?
val = ((Parent)c).field1;

//how do I access f1 and f2?
Parent.f1 = 0;
Child.f1 = -1; //Child’s
Child.f2 = 100; //Parent’s \(\rightarrow\) same as Parent.f2
Parent p = new Child();
//static methods are just like static fields:
Parent.foo();
Child.foo();

//instance method can use static dispatch (C++, C#)
p.bar()
((Parent)p).bar();
((Child)p).bar();

class Parent {
    ...
    static void foo() {...}
    int bar() {...}
}
class Child
    static void foo() {...}
    int bar() {...}
}
Parent p = new Child();
//static methods are just like static fields:
Parent.foo();  //same as p.foo() if type(p) == Parent
Child.foo();  //same as p.foo() if type(p) == Child

//instance method can use static dispatch (C++, C#)
p.bar();       //Parent's bar
((Parent)p).bar(); //Parent's bar
((Child)p).bar();  //Child's bar
Parent c = new Child();

//instance method can use **dynamic** dispatch (Java, C++ and virtual, C# and virtual key word, all dynamically typed languages)

c.foo();
c = new Parent();
c.bar();
((Child)c).foo();
c = new Child();
((Parent)c).bar();

```java
class Parent {
    ...
    void foo() {...}
    int bar() {...}
}
class Child
    void foo() {...}
    int bar() {...}
```
Parent c = new Child();
//instance method can use **dynamic** dispatch (Java, C++ and virtual, C# and virtual key word, all dynamically typed languages)
c.foo(); //Child's foo
c = new Parent();
c.bar(); //Parent's bar
(Child)c).foo(); //dynamic check then Parent's foo
c = new Child()
(Parent)c).bar(); //Child's bar()