

## Agenda

- upcasting
- downcasting
- instanceof
- Object class
  - toString
  - equals
- ~~abstract method~~
- ~~abstract class~~
- interface
- ~~marker interface~~

## Upcasting

- Assigning sub-class reference to a super-class reference.
- Sub-class "is a" Super-class, so no explicit casting is required.
- Using such super-class reference, only super-class methods inherited into sub-class can be called. This is "Object slicing".
- Using such super-class reference, super-class methods overridden into sub-class can also be called.

## Downcasting

- Assigning super-class reference to sub-class reference.
- Every super-class is not necessarily a sub-class, so explicit casting is required.

```
Person p1 = new Employee();  
Employee e1 = (Employee)p1; // down-casting - okay - Employee reference  
will point to Employee object  
  
Person p2 = new Person();  
Employee e2 = (Employee)p2; // down-casting - ClassCastException - Employee  
reference will point to Person object
```

## Polymorphism

- poly = Many , morphism = Forms
- It has two types
  1. compile time
    - implemented using method overloading
    - Compiler can identify which method to be called at compile time depending on types of arguments. This is also referred as "Early binding".
  2. Runtime - implemented using method overriding - The method to be called is decided at runtime depending on type of object. This is also referred as "Late binding" or "Dyanmic method dispatch".

## instanceof operator

- Java's instanceof operator checks if given reference points to the object of given type (or its sub-class) or not. Its result is boolean.
- Typically "instanceof" operator is used for type-checking before down-casting.

```
Person p = new SomeClass();
if(p instanceof Employee) {
    Employee e = (Employee)p;
    System.out.println("Salary: " + e.getSalary());
}
```

## Object class

- Non final and non-abstract class declared in java.lang package.
- In java, all the classes (not interfaces) are directly or indirectly extended from Object class.
- In other words, Object class is ultimate base class/super class hierarchy.
- Object class is not inherited from any class or implement any interface.
- It has a default constructor.  
Object o = new Object();
- Object class methods (read docs)
  - public Object();
  - public native int hashCode();
  - public boolean equals(Object);
  - protected native Object clone() throws CloneNotSupportedException;
  - public String toString();
  - protected void finalize() throws Throwable;
  - public final native Class<?> getClass();
  - public final native void notify();
  - public final native void notifyAll();
  - public final void wait() throws InterruptedException;
  - public final native void wait(long) throws InterruptedException;
  - public final void wait(long, int) throws InterruptedException;

## toString() method

- it is a non final method of object class
- To return state of Java instance in String form, programmer should override toString() method.
- The result in toString() method should be a concise, informative, and human-readable.
- It is recommended that all subclasses override this method.

## equals() method

- It is non final method of object class
- To compare the object contents/state, programmer should override equals() method.
- This equals() must have following properties:
  - Reflexive: for any non-null reference value x, x.equals(x) should return true.
  - Symmetric: for any non-null reference values x and y, x.equals(y) should return true if and only if y.equals(x) returns true.
  - Transitive: for any non-null reference values x, y, and z, if x.equals(y) returns true and y.equals(z) returns true, then x.equals(z) should return true.
  - Consistent: for any non-null reference values x and y, multiple invocations of x.equals(y) consistently return true or consistently return false, provided no information used in equals comparisons on the objects is modified.
- For any non-null reference value x, x.equals(null) should return false.
- It is recommended to override hashCode method along when equals method is overridden.

## Labwork

- Do the implementation of Shape class example in c++