# CMSC 132
# Week2 Lab1

Comparable vs Comparator

clone()

finalize()

## Student Class

```
public class Student {
      String first;
      String last;
      int id;

      public Student(String first, String last, int id) {
            this.first = first;
            this.last = last;
            this.id = id;
      }
}
```

## Comparable vs Comparator

- Both are interfaces
- `Comparable` defines the natural ordering for class objects
  - Has only one method `compareTo`
- Comparison by lastname for `Student` class

```
public class Student implements Comparable<Student>{
      ……
      @Override
      public int compareTo(Student o) {
            return this.last.compareTo(o.last);
      }
}
```

## Comparable vs Comparator

- How can we add more than one comparison method ?
  - Define a `Comparator` class for each comparison method
  - Has only one method `compare`

```
public class StudentComparatorByID implements
Comparator<Student>{

     @Override
     public int compare(Student o1, Student o2) {
          return o1.id - o2.id;
     }

}
```

## Comparable vs Comparator – Example

```
Student s1 = new Student("John", "Locke", 5);
Student s2 = new Student("Mike", "Nickolas", 3);

s1.compareTo(s2);       // -ve
s2.compareTo(s1);       // +ve

StudentComparatorByID comp = new StudentComparatorByID();
comp.compare(s1, s2);   // +ve
comp.compare(s2, s1);   // -ve
```

## Object’s Class Methods

- equals()
- toString()
- clone()
- finalize()
- … etc

## clone()

- Method's signature

```
protected Object clone() throws CloneNotSupportedException
```

- It is used to make copy of objects
- Example

```
        Student sCopy = (Student)s.clone();
```

- Default behavior throws an exception when class is not `Cloneable`
  - Unless you override it

## clone() – Student Class

```
@Override
protected Object clone() throws CloneNotSupportedException {
      return new Student (first, last, id);
}
```

- Example

```
        Student sCopy = (Student)s.clone();
```

- Why don't we use the following ?

```
                    Student sCopy = s;
```

## finalize()

- Garbage Collector (gc) can delete objects with no references
  - Orphaned object
- Before deleting an object, gc calls the object's `finalize` method
  - For any cleanup required
  - Do anything special
- Method's signature

```
protected void finalize() throws Throwable {
```

## finalize() – Student Class

```
@Override
protected void finalize() throws Throwable {
        System.out.println("I'm gonna die :(");
}
public static void foo(){
        // Just creates an object and terminates
        Student s = new Student("John", "Luke", 5);
}
public static void main(String[] args) {
        foo();
        // explicit call for gc to run (runs automatically by default)
        System.gc();
}
```

# Clicker Quiz

06/15/2017

1. Suppose we have a class Person that overrides the clone() method

```
Person p = new Person();
Person pCopy = p.clone();
```

A. Compiles with a warning

B. Doesn’t compile

C. Compiles but would cause runtime error.

D. Works just fine.

1. Suppose we have a class Person that overrides the clone() method

```
Person p = new Person();
Person pCopy = p.clone();
```

A. Compiles with a warning
B. Doesn’t compile
C. Compiles but would cause runtime error.
D. Works just fine.

2. True or false. The default implementation of the clone() method performs a shallow copy only.

A. True
B. False

2. True or false. The default implementation of the clone() method performs a shallow copy only.

A. True
B. False

3. True or false. The code inside finalize() will not execute unless we explicitly call System.gc().

A. True

B. False

3. True or false. The code inside finalize() will not execute unless we explicitly call System.gc().

A. True

B. False

4. True or false. If we a comparator StudentComparatorByID, that compares two Student objects, then the Student class cannot implement the Comparable interface or else we'll get a conflict between the compareTo method defined by the Comparator and Comparable interfaces.

A. True
B. False

4. True or false. If we have a comparator StudentComparatorByID, that compares two Student objects, then the Student class cannot implement the Comparable interface or else we'll get a conflict between the compareTo method defined by the Comparator and Comparable interfaces.

A. True
B. False

## TODO - For the Time class

- A Comparator class (compare only by hours)
- A clone method
- A finalize method to the Time class and see the effect of running System.gc().
- An equals() method

- Try to work with the same partner
- Feel free to ask
- Write some student test cases for these features