Midterm 2

Thursday, 5 to 7pm
What results are printed out?

Double d = new Double(5.2);
System.out.println( “result:” + ( d > 5));

double [][] d = { {5.6, 2.3},{2.7, 4.5}};
System.out.println( “d[1][1]:” + d[1][1]);
What is wrong?

```java
public class C {
    private final int b;
    C() {
        b = 1;
    }
    public void setB( int newB) {
        b = newB;
    }
}
```
New Material

Inheritance and Polymorphism (chapter 11)

Exception Handling (chapter 12)

Text I/O (chapter 12)

Abstract Classes (chapter 13)

Interfaces (chapter 13)
Inheritance & Polymorphism
Assume:
KitchenTool kt = new Microwave();
CuttingTool ct = new CuttingTool();

1. Is kt instanceof Object?
2. Is kt instanceof CookingTool?
3. Is kt instanceof KitchenTool?
4. Is kt instanceof Oven?
5. Is kt instanceof Microwave?
6. Is ct instanceof CookingTool?
7. Is ct instanceof KitchenTool?
8. Is ct instanceof CuttingTool?
Assume:
Object o = new Microwave();
CuttingTool ct = new CuttingTool();
Knife k = new Knife();

If method turnOn() is defined in CookingTool,
1. can ct invoke this method?
2. can o invoke this method?
3. Is ((CookingTool)o).turnOn() legal?

If method cut() is defined in CuttingTool
4. is ct.cut() legal?
5. is k.cut() legal?
class Kitchen {
    public static void main(String[] args) {
        Object o = new CookingTool();
        Oven oven = (Oven)o;
    }
}

class KitchenTool {}
class CookingTool extends KitchenTool {}
Hmmm...

Stove s = new Stove();
Object o = new CookingTool();
KitchenTool kt = new KitchenTool();
Knife k = new Knife();

Assume CookingTool has an equals method, Which equals method will
1. s.equals( o) call?
2. o.equals( s) call?
3. kt.equals( o) call?
4. is s.toString() legal?
5. is o.toString() legal?
6. is k.toString() legal?
1. Will this compile?
2. Which equals method overrides Object’s?
3. Are the equals methods overloaded?
4. s1.equals( o) which method is called?
5. o.equals( s1) which method is called?
6. s1.equals( s2) which method is called?
7. Is the explicit cast an upcast or downcast?
8. Is the explicit cast safe? If not, what would safe syntax be?
What is the error?

```java
package a;
public class A {
    int a;
}

package b;
import a.A;
public class B extends A {
    int b = super.a;
}
```
What is the error?

package a;
public final class A {
    int a;
}

package b;
import a.A;
public class B extends A {
    int b = super.a;
}
What is the bug?

class IntList {
    private ArrayList<Integer> list = new ArrayList<>();
    public void add( int i) {
        list.add( new Integer(i));
    }
    public void add2( int i, int j) {
        add( i);  add( j);
    }
}

//TEST CODE
TrackCount tc = new TrackCount();
tc.add(3); tc.add2( 2, 4);
System.out.println( "tc.count=" + tc.count);

class TrackCount extends IntList {
    int count = 0;
    public void add( int i) {
        this.count++;
        super.add( i);
    }
    public void add2( int i, int j) {
        this.count += 2;
        super.add2( i, j);
    }
}
What is the bug?

```java
public final class Sub extends Super {
    private final Date date;
    Sub() {
        date = new Date();
    }
    @Override
    public void overrideMe() {
        System.out.println(date);
    }
}

public class Super {
    public Super() {
        overrideMe();
    }
    public void overrideMe() {
    }
}

//Test Code
Sub sub = new Sub();
sub.overrideMe();
```

From Joshua Bloch, Effective Java
Exception Handling

Runtime Errors

Unchecked Exceptions

Error - internal system errors

RuntimeException

Checked Exceptions

Exception

NullPointeException
NumberFormatException
ClassCastException
IndexOutOfBoundsException
...

IOException
SQLException
MalformedURLException
...
RuntimeExceptions - Programming Errors

System.out.println( 1 / 0);

int[] list = new int[4];
System.out.println( list[4]);

String s = “abc”;  
System.out.println( s.charAt(3));

Object o = new Object();
String d = (String)o;

Object o = null;
o.toString();
RuntimeExceptions - Programming Errors

System.out.println(1 / 0); //ArithmeticException

int[] list = new int[4];
System.out.println(list[4]);  //ArrayIndexOutOfBoundsException

String s = "abc";
System.out.println(s.charAt(3));  //StringIndexOutOfBoundsException

Object o = new Object();
String d = (String)o;  //ClassCastException

Object o = null;
o.toString();  //NullPointerException
Javadocs

http://docs.oracle.com/javase/7/docs/api/java/lang/RuntimeException.html

Exception in thread "main" java.lang.ClassCastException: java.lang.Object cannot be cast to java.lang.String
   at Test.method2(Test.java:5)
   at Test.method1(Test.java:9)
   at Test.main(Test.java:27)
Checked Exceptions

Not Error or RuntimeException

Compiler forces programmer to check and deal with them.

//declare method throws them

public void myMethod() throws IOException { }
Catching Exceptions

try {

    some statements;

} catch ( Exception1 e) {

} catch ( Exception2 e) {

}
Throwing Exceptions

```java
public void myMethod(int i) throws IllegalArgumentException {
    if (i < 0) {
        throw new IllegalArgumentException("i cannot be negative");
    }
}

//IllegalArgumentException is a RuntimeException
```
What happens when...

```java
try {
    statement1;
    statement2;  //throws an exception
    statement3;
} catch( Exception1 e) {

} catch( Exception2 e) {

}

statement4;

//will statement 3 be executed?
//if exception is not caught will statement 4 be executed?
//if the exception is caught will statement 4 be executed?
```
What happens when...

```java
try {
    statement1;
    statement2;
    statement3;
} catch( Exception1 e) {

} finally {
    statement4;
    statement4;
}
```