OBLIG 1 – Programmering 2

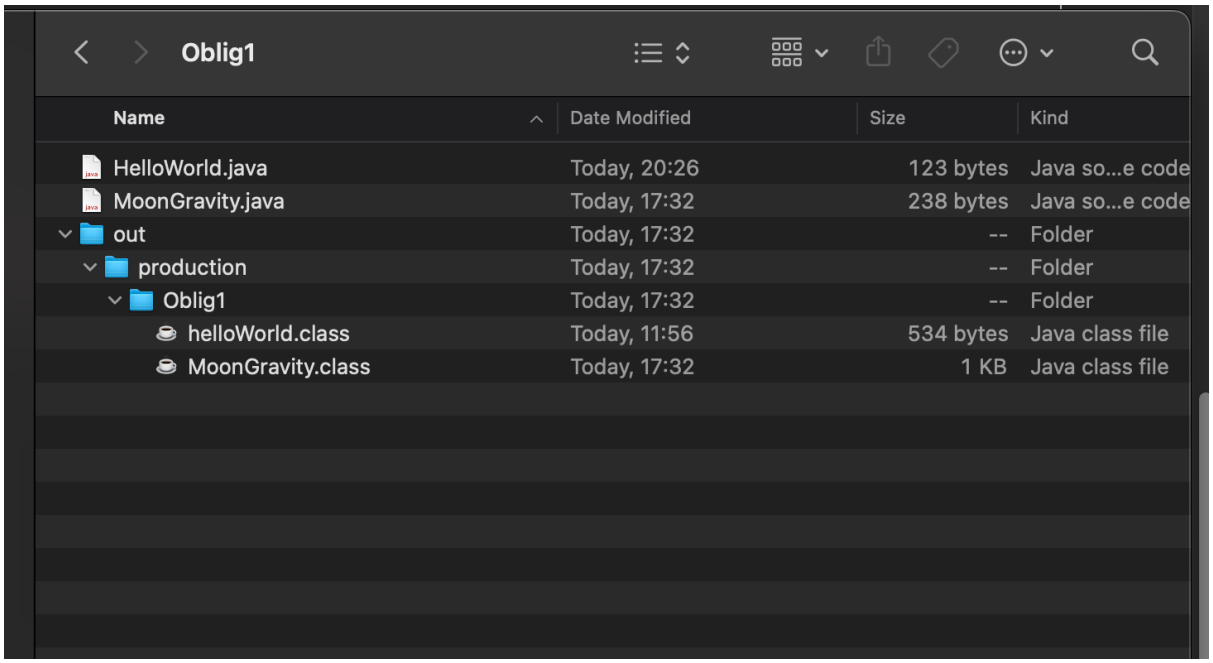
Oppgave 1.1 – *Forklar hva JRE (Java Runtime Environment) og JDK (Java Development Kit) er.*

Svar: **JRE**, Java Runtime Environment, er software vi må installere for å kunne kjøre Java-programmer på maskinen. Det er det som kan kalles «runtime» delen av Java. I JRE finnes det som heter Java Virtual Machine, JVM. Dette er den delen av software som kjører koden. I JRE finnes også bibliotek/ressurser (Libraries) som er nødvendige for Java-språket, som f.eks. java.lang. – package som har klasser som Object, System, String, osv. Inneholder med andre ord pre-programmerte verktøy, datastrukturer og algoritmer, som består av mange «pakker» (packages).

JDK, Java Development Kit, har alle komponenter, all funksjonalitet som JRE har, men har i tillegg verktøy for utvikling, koding i Java. Dette er nødvendig for å kunne utvikle programmer i Java. I JDK finnes Java Compiler – det som oversetter Javakoden til bytecode som kan kjøres av den virtuelle maskinen (JVM). I JDK finnes også Java Debugger – som hjelper med å finne feil/bugs i programmet.

Oppgave 1.2 - Forklar flyten av hvordan java-applikasjoner blir bygget og kjørt.
Knytt også forklaringen din opp mot terminal-kommandoene, *javac* og *java*.

Svar: For å bygge og kjøre java-applikasjoner starter vi med å skrive kode, filer (klasser) som har extension-navnet .java. For å kjøre et enkelt program som f.eks «Hello World» kan vi bruke «javac»-kommandoet i terminalen. Javac er compilerdelen av JDK og er det som oversetter javakode til bytecode. Det som skjer når dette kjøres er at compilern lager .class filer som inneholder bytecode, sånn som er vist i screenshot herunder:



(Screenshot fra egen maskin)

Når vi kjører programmet via IntelliJ så er det som at javac og java (med filnavn) kommando kjøres. IntelliJ kompilerer koden automatisk.

Når vi har kjørt javac-kommando og kompilert koden, kjører vi programmet med java, f.eks. java HelloWorld, outputen i terminalen blir da, Hello World! Her kommer java virtual machine (JVM) inn i bildet. Denne leser og kjører bytekoden som finnes i .class-filen. (det som har blitt kompilert med javac). JVM ser etter main-metoden, som er startpunktet for alle javaprogrammer.

Oppgave 1.3 – *Forklar forskjellen på Compile-time og Run-time errors. Gi noen eksempler*

Svar: **Compile-time errors** er feil som skjer under kompileringen av koden, altså når java compilern (javac) skal prøve å konvertere koden, det vi skriver som .java filer, til .class filer. Det kan være så enkelt som syntax-feil som manglende semikolon, manglende parenteser, eller at man har assignet en int-verdi til string-variabel. Det kan også være at man bruker keywords (de spesielle nøkkelord som er programmert inn i java) på feil måte. For å illustrere så kan det være så simpelt som dette: int num1 = «Hei». En string kan ikke assignes til en int-variabel, som nevnt før.

Run-time errors er feil som skjer under selve kjøringen (run time) av programmet, etter at det har blitt kompilert korrekt. Det kan være feil som brukeren gjør – sette inn feil verdier ved input f.eks – hvis ikke dette håndteres korrekt vil det «krasje» - run-time error. Eksempelvis som å prøve å dele noe på 0

Oppgave 1.4 - *Forklar forskjellen på en klasse og et objekt. Vis gjerne et lite eksempel på hvordan man definerer en klasse, og oppretter et objekt*.

En klasse er som en oppskrift vi bruker til å lage versjoner/instanser av klassen. Disse kaller vi objekter. Klassen er altså oppskriften som vi lager unike instanser av som alle kan ha sine unike egenskaper. F.eks kan vi lage klassen Student slik:

```java
//Oppretter og definerer en klasse her:
new *
public class Student {
    2 usages
    private String navn;
    2 usages
    private int studentNummer;


    1 usage  new *
    public Student(String navn, int studentNummer) {
        this.navn = navn;
        this.studentNummer = studentNummer;
    }
```

```java
    //Oppretter et objekt av Studentklassen her:
    new *
    public static void main(String[] args) {
        Student student1 = new Student( navn: "John",  studentNummer: 1234);
        System.out.println("Navn: " + student1.navn + ", Studentnummer: " + student1.studentNummer);
    }

}
```