Oppgave 1.1

JRE står for Java Runtime Enviornment og er nødvendig for å kjøre Java-applikasjoner. JRE inneholder flere ting, blant annet JVM eller Java Virtual Machine. Dette er en maskin som utfører Java kode som er blitt kompilert. Denne koden heter bytekode og er et mellomsteg mellom den lesbare Java koden og maskinkoden. At koden er kompilert betyr at den er tolket/oversatt til enten en lesbar form for datamasker eller for oss mennesker.

JRE inneholder også ulike Java biblioteker og klasser som gir oss funksjonalitet som ofte brukes. Disse gir oss tilgang til biblioteker med standardfunksjoner, slik at vi ikke trenger å kode alt fra bunnen av. JRE inneholder også alt av ressursfiler og annet nødvendig innhold for å støtte kjøringen av Java apper.

JRE gir oss rett og slett et miljø hvor Java kan kjøre uavhengig av plattformen brukeren er på. Dette betyr at man kan kode Java på Windows, Linux og MacOS og bytte imellom disse uten å kompilere på nytt.

JDK eller Java Development Kit er en utviklingspakke for Java, som inneholder alt som er nødvendig for å utvikle Java apper. JDK inneholder mye av det samme som JRE, hvor hovedforskjellen egentlig handler om debuggingsmuligheter.

For å oppsummere, hvis man bare ønsker å kjøre ferdige Java-programmer, er JRE tilstrekkelig. Men hvis man vil utvikle Java-programmer, trenger man JDK. JDK inkluderer alt som JRE har, i tillegg til utviklingsverktøyene som blant annet debuggingmuligheter.

Oppgave 1.2

Flyten i et java program eller java app er automatisert av IDE-er som VsCode eller Intellij IDEA, hvor den kompilerer og gjør alt sammen for deg, men under vises fremgangsmåten hvis man skal gjøre det manuelt, og hvordan IDE-er som VsCode gjør det.

1. Man må først skrive Java koden sin i en teksteditor eller IDE som f.eks. VsCode eller intellij.

2. Etter å ha skrevet koden må den kompileres/oversettes. Dette kan gjøres ved å skrive følgende i terminalfeltet/kommandofeltet: **«javac 'Filnavn'.java».** (uten fnutter)
3. Etter at koden er kompilert, oppstår det en bytekode. Dette er en mellomliggende kode som kan kjøres på alle plattformer som har en JVM.
4. Nå som man har bytekoden kan man kjøre den ved hjelp av følgende kommando i terminalfeltet/kommandofeltet: **«java 'Filnavn'»** (uten fnutter).
5. Når man kjører programmet med kommandoen over, lastes bytekoden inn i JVM som tolker og utfører koden som tidligere ble kompilert
6. Resultatet av koden blir presentert i konsollområdet.

Oppgave 1.3

Compile-time errors oppstår når man kompilerer koden, og før programmet faktisk kjører. Disse feilene hindrer kompilatoren å generere kjørbar kode, på grunn av syntaks- og strukturfeil. Compile-time errorer må derfor rettes opp i før man får kjørt programmet. Under er et eksempel på en error. Denne oppstår pga. int x blir satt som String og ikke integer.

```
1  public class Eksempel {
2      public static void main(String[] args) {
3          int x = "Hello";
4          System.out.println(x);
5      }
6  }
```

Run-time errors oppstår når programmet kjører (etter kompileringen). Dette er feil som skyldes logikkfeil, ugyldige operasjoner og uventende situasjoner. Hvis man har slike problemer, vil programmet avbrytes eller oppføre seg rart. Under er et eksempel hvor man prøver å hente ut et tall fra en index som ikke finnes, og man får derfor feilen «ArrayIndexOutOfBoundsException**».**

```
1  public class Eksempel {
2      public static void main(String[] args) {
3          int[] numbers = {1, 2, 3};
4          System.out.println(numbers[3]);
5      }
6  }
```

Oppgave 1.4

En metode en samling av instrukser som skal utføre spesifikke oppgaver. En metode ligger inne i en klasse og brukes til å definere funksjonaliteten til objektene. En metode kan ha parametere som tar imot verdier, men dette er ikke alltid nødvendig. Under er et eksempel på en metode som skriver ut «Hei fra metoden!»

```
1  public class EksempelKlasse {
2      public void siHei() { // Dette er en metode
3          System.out.println("Hei fra metoden!");
4      }
5  }
```

En klasse brukes generelt sett til å organisere kode og definere objekter, mens metoder er ansvarlige for å utføre spesifikke handlinger. En klasse er egentlig en blueprint eller oppskrift for elementer i programmet.