Matrikelnummer:

Aufgabe 1: Quickies (25 Punkte)

a) Wird das folgende Programm von einem Java-Übersetzer ohne Beanstandungen übersetzt? Falls nicht, warum? Begründen Sie Ihre Antwort! (3 Punkte)

```
class Peter {
    void Arnd(String Wilfried) {
      /* ... */
    }

}

class Joerg extends Peter {
    private void Arnd(String Monika) {
      /* ... */
    }
}
```

b) Beschreiben Sie möglichst genau, aus welchen Teilen die folgende Java-Anweisung aufgebaut ist und was diese Teile bedeuten!
`System.err.print();`

(3 Punkte)

c) Zeichnen Sie das Objektgeflecht, das entstanden ist, wenn die Ausführung der main-Methode die markierte Stelle erreicht hat! (5 Punkte)

```
class Geflecht {
    Geflecht a,b,c;

    public synchronized static void main(String[] Ute) {
        Geflecht u = new Geflecht();
        Geflecht v = new Geflecht();
        Geflecht w = new Geflecht();
        u.a = v;
        v.a = w;
        w.a = u.a;
        v.b = u;
        u.b = v;
        v.c = v.a;
        w.c = u;
        u.c = v.b;
        /* Markierung */
    }
}
```

d) Betrachten Sie das folgende Java-Programm:

```
class A {
    void m() {
        if(this instanceof A) System.out.println("Ich bin ein A Objekt");
        if(this instanceof B) System.out.println("Ich bin ein B Objekt");
    }
}

class B extends A {

    void m() {
      super.m();
    }

    public static void main(String[] s) {
        A a = new B();
        a.m();
    }
}
```

Was wird bei der Ausführung der main-Methode ausgegeben? Begründen sie Ihre Antwort!
(3 Punkte)

e) Was wird ausgegeben, wenn die main-Methode des folgenden Programms ausgeführt wird? Begründen Sie ihre Antwort!
(3 Punkte)

```
class K {
    K() { System.out.println("K"); }

    public static void main(String[] oho) { new M(); }
}

class L extends K {
    L() { System.out.println("L"); }
}

class M extends L {
    M() { System.out.println("M"); }
}
```

f) Schreiben Sie ein kleines Programmbeispiel, bei dessen Ausführung dynamisch gebunden wird! Erklären Sie genau, an welcher Stelle der Programmausführung dynamisch gebunden wird! Ihr Beispiel soll übersetzbar und ausführbar sein. (4 Punkte)

g) Schreiben Sie ein kleines Programmbeispiel, an dessen Ausführung man sehen kann, dass der Attributzugriff in Java nicht dynamisch gebunden wird! Ihr Beispiel soll übersetzbar und ausführbar sein.
(4 Punkte)

Aufgabe 2: Simulation von dynamischer Bindung (40 Punkte)

In dieser Aufgabe sollen Sie die dynamische Methodenbindung explizit ausprogrammieren. Dazu werden dynamisch gebundene Methoden nach dem folgenden Muster ersetzt. M bezeichne im Folgenden eine dynamisch gebundene Methode in der Klasse C:

- M wird durch eine `static`-Methode MS ersetzt, der der implizite Parameter als erster Parameter explizit übergeben wird.
- Da der Typ des impliziten Parameters von M gleich C oder ein Subtyp von C ist, hat der erste Parameter von MS den Typ C.
- Wird die Methode MS aufgerufen, so muss anhand des Typs des impliziten Parameters entschieden werden, ob die Methode MS ausgeführt werden soll, oder die Methode eines der Subtypen.
- Soll die Methode eines der Subtypen ausgeführt werden, so ist der Methodenaufruf an die Methode der entsprechenden Subtypen zu delegieren, für den die gleiche Ersetzung erfolgt, wie für die Methode M (natürlich nur, falls Subtypen existieren!). Dabei ist der implizite Parameter (der ja explizit übergeben wird) auf den entsprechenden Subtypen zu casten. MS soll Methodenaufrufe nur an direkte Subtypen von C delegieren.
- Die Aufrufstellen der so behandelten Methoden sind entsprechend anzupassen.

Aufgaben:

a) Ersetzen Sie im unten angegebenen Programm die Implementierungen der dynamisch gebundenen Methoden und Methodenaufrufstellen durch `static`-Methoden nach dem obigen Muster, so dass bei einer Ausführung die gleiche Ausgabe wie die unten angegebene erzeugt wird!
(25 Punkte)

b) Mit Techniken wie der obigen könnte man mit imperativen Programmiersprachen objektorientiert programmieren. Welchen entscheidenden Nachteil hat die oben angegebene Technik? Denken Sie dabei an Aspekte wie Wiederverwendung, Frameworks und Typinformationen!
(5 Punkte)

c) Ebenso wie man innerhalb jeder ersetzten Methode wie oben (also dezentral) die dynamische Bindung explizit ausprogrammieren kann, könnte man die dynamische Bindung auch direkt beim Methodenaufruf auflösen. Dazu kann man eine `static`-Methode schreiben, die anstelle der dynamisch gebundenen Methode mit dem impliziten Parameter und den restlichen Parametern aufgerufen wird und die die dynamische Methodenauswahl implementiert (d.h. die dynamische Bindung wird zentral in einer Methode aufgelöst).

Geben Sie analog zu Aufgabenteil a) ein Muster an, mit dem man dynamisch gebundene Methoden, wie gerade beschrieben, explizit ausprogrammieren kann. Beschreiben Sie, welchen Vorteil diese Technik im Vergleich mit der in a) beschriebenen Technik hat! Wenden Sie die Technik auf das Beispielprogramm an!
(10 Punkte)

```
class Dynamic {
    String m() {
        return "m in Dynamic";
    }

    public static void main(String[] s) {
        Dynamic[] d = new Dynamic[4];
        d[0] = new Dynamic();
        d[1] = new S1();
        d[2] = new S2();
        d[3] = new S3();
        for(int i=0;i<d.length;i++) System.out.println(d[i].m());
    }
}

class S1 extends Dynamic  {
    String m() {
       return "m in S1";
    }
}

class S2 extends Dynamic  {
    String m() {
        return "m in S2";
    }
}

class S3 extends S2 {
    String m() {
        return "m in S3";
    }
}
```

Das Program erzeugt bei der Ausführung der `main`-Methode die folgende Ausgabe:

```
m in Dynamic
m in S1
m in S2
m in S3
```

Aufgabe 3: Threads (35 Punkte)

a) Jedes `Thread`-Objekt hat eine `public void run()` und eine `public void start()`-Methode. Beschreiben Sie die unterschiedliche Bedeutung der beiden Methoden!
(5 Punkte)

b) In dieser Teilaufgabe sollen Sie einen sogenannten ThreadPool implementieren. Ein ThreadPool ist ein Objekt, das eine gewisse Anzahl von Threads zur Berechnung von Aufgaben zur Verfügung stellt. Man kann einem ThreadPool-Objekt eine zu berechnende Aufgabe übertragen. Das ThreadPool-Objekt speichert die Aufgabe und lässt sie von einem seiner Threads berechnen, sobald ein Thread frei ist. Gegeben sei das folgende Programmfragment:

```
class ThreadPool {
    ...
    public  ThreadPool(int m) {
      ...
    }

    ... public void doJob(Runnable r) {
      ...
    }

    ... public void  waitUntilComplete() {
      ...
    }
}
```

Implementieren Sie die Methoden des Klassenfragmentes so, dass die folgenden Eigenschaften gelten:
(20 Punkte)

- Der Konstruktor der Klasse `ThreadPool` erzeugt ein `ThreadPool`-Objekt mit der als `int`-Wert übergebenen Anzahl an Threads.
- Mit der Methode `doJob` kann einem ThreadPool ein Job zum Berechnen übertragen werden. Jobs sind vom Schnittstellentyp `Runnable`.
- Die Methode `waitUntilComplete` blockiert einen aufrufenden Thread so lange, bis alle Jobs des ThreadPools abgearbeitet sind.

c) Gegeben sei das folgende Fragment einer Implementierung des Quicksort-Algorithmus, die parallelisiert werden soll. Die `partition(int l, int r)`-Methode sorgt dafür, dass folgendes gilt:

1. Das i-te Element befindet sich an seinem endgültigen Platz im Feld,
2. $\forall(k : l \leq k \leq i-1) : a[k] \leq a[i]$ und
3. $\forall(k : i+1 \leq k \leq r) : a[k] \geq a[i]$.

Dabei kann ausgenutzt werden, dass die rekursiven Aufrufe von Quicksort auf verschiedenen Teilen des zu sortierenden Feldes arbeiten.

```
class Quicksort {
    int a[],l,r;

    public static void main(String[] sv) {
        int f[] = { 1,5,3,4,5,6,7,6,5,4,3,4 };
        new Quicksort(f,0,f.length-1);
        System.out.println("Sortiertes Feld:");
        for(int i=0;i<f.length;i++) System.out.println(f[i]);
    }

    Quicksort(int a[], int l, int r) {
        this.a=a;
        this.l=l;
        this.r=r;
        run();
    }

    public void run() {
        int i;
        if(r>l) {
           i = partition(l,r);
           new Quicksort(a,l,i-1);
           new Quicksort(a,i+1,r);
        }
    }

    static int partition(int l, int r) {
        // Implementierung wird an dieser Stelle nicht benötigt
    }

}
```

Aufgabe: Verändern Sie die gegebene Quicksort-Implementierung so, dass die rekursiven Aufrufe als Jobs von einem ThreadPool parallel abgearbeitet werden! (Die Methode `partition` brauchen Sie nicht zu implementieren.) (10 Punkte)